Advancing Deep Active Learning & Data Subset Selection: Unifying Principles with Information-Theory Intuitions

Andreas Kirsch

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2023

To my family and friends.

In particular:
my parents, for their support;
Armin, for his advice; and
Alex, in his memory.

Abstract

At its core, this thesis aims to enhance the practicality of deep learning by improving the label and training efficiency of deep learning models. To this end, we investigate data subset selection techniques, specifically active learning and active sampling, grounded in information-theoretic principles. Active learning improves label efficiency, while active sampling enhances training efficiency.

Supervised deep learning models often require extensive training with labeled data. Label acquisition can be expensive and time-consuming, and training large models is resource-intensive, hindering the adoption outside academic research and "big tech."

Existing methods for data subset selection in deep learning often rely on heuristics or lack a principled information-theoretic foundation. In contrast, this thesis examines several objectives for data subset selection and their applications within deep learning, striving for a more principled approach inspired by information theory.

We begin by disentangling epistemic and aleatoric uncertainty in single forward-pass deep neural networks, which provides helpful intuitions and insights into different forms of uncertainty and their relevance for data subset selection. We then propose and investigate various approaches for active learning and data subset selection in (Bayesian) deep learning. Finally, we relate various existing and proposed approaches to approximations of information quantities in weight or prediction space.

Underpinning this work is a principled and practical notation for information-theoretic quantities that includes both random variables and observed outcomes. This thesis demonstrates the benefits of working from a unified perspective and highlights the potential impact of our contributions to the practical application of deep learning.

Acknowledgements

To my family and friends: thank you for your unwavering support throughout my DPhil journey. In particular, I am grateful to my parents, who have supported me in so many ways over the years; to my cousin, who shared with me the ups and downs of pursuing a doctorate at the same time in Romania; and my old friends in my hometown in Germany: Michi, Daniel, and Marina. I hope to see you all soon.

Special thanks go to my old friend, Armin Krupp, who pointed me to the exciting world of machine learning at the University of Oxford. Your positivity and support have been a driving force in my academic and professional journey. And just as much am I grateful to Daniel and Martin for sticking with me since our time together in Munich.

I also owe much to the many friends who have made my time in Oxford worthwhile: especially, Alessandro, Kevin, Alessio, Maja, Jonas, Charlotte, Lorika, Bastiaan, Charles, Prateek, Luisa, Jan, Clemens, Fabian, Ana, Rob, Brenda, Neil, Theo, Kayla, Anna, Aleksandra, Fiona, Serban, Stefan, Patricia, Olmo, Carli, Simone, Mar, John, Ed, Josh, Shad, Sam & Tim; your friendship has been invaluable, whether on (coffee) walks, at shared dinners and co-working sessions, in Italy, singing karaoke, surviving the pandemic, hiking up the tallest mountains or in the Cotswold, visiting me, advising me, partying or going to formals together. The same goes for the Society Cafe regulars & baristas over the last five years: Nadia, Dom, Laura, Em, Amina, Will, Johnny, Marie, Kat, Eleanor, Ricardo, Shae, Ruth, Lauren & Noor; thank you for being so much fun every day; and to my DnD group: Lasya, Cyril, Mihir, Nico, Ned & Luca; my experience would not have been the same without you and Gogol.

I am very grateful to the AIMS CDT program and its incredible administrator, Wendy Poole, for providing funding and support throughout my studies and generally being a force for good. I am also very lucky to be a student of Exeter College which I thank from my heart for their support and nurturing environment—an environment which starkly contrasts with my earlier experiences at Kellogg College.

I am also grateful to my collaborators, especially my joint co-authors, for their support and feedback: Tom Rainforth for his mentorship and invaluable advice on research and writing—meeting in University Parks during the pandemic with a small

whiteboard to discuss information theory and transductive acquisition functions is something I will always remember; my external collaborators, including Dustin Tran, Jasper Snoek, Frédéric Branchaud-Charron, Parmida Atighehchian, and Uri Shalit; and my OATML collaborators, Joost van Amersfoort, Jishnu Mukhoti, Sebastian Farquhar, Freddie Bickerforth Smith, Muhammed Razzak, Andrew Jesson, Jannik Kossen, Clare Lyle, Jan Brauner, Sören Mindermann, Panagiotis Tigas, and Aidan Gomez—your expertise, encouragement, and patience have been invaluable to me and this thesis impossible without you. My thanks goes to the other members of OATML, too, for their support and feedback: especially Lisa Schut, Tim Rudner, Milad Alizadeh, and Lewis Smith.

I also want to thank my supervisor, Yarin Gal, for supporting my DPhil journey and providing regular feedback. Due to the circumstances of our collaboration, I learned to take full ownership of my work, submitting my final paper (§10) as a sole author, and faced with artificial time constraints, I completed this thesis within an accelerated three-week timeframe. I believe I have become a more independent and self-aware scholar in many parts thanks to him.

While I have strived for research integrity as a guiding principle, it has tested my resilience and highlighted the delicate balance between being critical and patient in scholarly conversations. Being referred to as an 'unhinged reviewer 2' by an NYU professor (and former PhD colleague of my supervisor) for engaging in scholarly discourse and highlighting inconsistencies, while difficult, taught me about assertiveness and perseverance. These experiences have enriched my understanding of the academic landscape, both in the face of criticism and in making thoughtful contributions. The encouragement and kindness of the broader academic community despite this have played a pivotal role in my growth as a scholar, and I'm grateful for the opportunity to learn this during my DPhil.

Finally, I want to thank all those who have provided feedback on my thesis, especially my assessors, Kevin Murphy and Mike Osborne, and the many anonymous conference reviewers who have helped shape my work.

While I have failed in my research at times, this thesis stands as a testament to the collective efforts, support, and encouragement of the remarkable individuals who have been part of my DPhil journey. My deepest and sincerest thanks to you all.

Andreas Kirsch, Oxford, 2023

Contents

When you can do nothing, what can you do?

1 Preliminaries

1.1 Introduction

Over a decade ago, the deep learning revolution began, making significant strides in various research fields, including areas once considered exclusive domains of human ingenuity and creativity. Despite its success, deep learning still faces challenges that hinder its deployment in more practical, everyday settings. This thesis aims to address some of these challenges and make deep learning more accessible by reducing the costs of gathering and labeling data, speeding up training, and doing so in a principled fashion.

Deep learning models have yet to gain a strong foothold outside big tech, maybe due to issues such as lack of interpretability, robustness, and generalization guarantees. However, history tells us that such concerns are unlikely to deter the use of a new technology. So, what does?

More practical concerns, such as the time-consuming and expensive processes of gathering and labeling high-quality data and training models, may also limit the deployment of deep learning. Research often overlooks the data pipeline, which frequently is the bottleneck that constrains model performance.

The primary objective of this thesis is to address these practical challenges and make deep learning more accessible by exploring how to reduce the cost of gathering and labeling data and how to speed up training. At the same time while maintaining a principled approach allows for a better understanding of trade-offs and connections between different methods.

In particular, this thesis focuses on *data subset selection* in the broader sense of active learning and active sampling for deep learning models, especially through the principled application of information theory. *Active learning*, like semi-supervised and unsupervised approaches, increases label efficiency. It does so by selecting the most informative samples to label from a *pool set* of unlabeled data, potentially significantly reducing the number of required labels for good model performance. Similarly, *active sampling* improves training efficiency by filtering the available training data to focus on the most informative samples for the model during training. The main research questions in this thesis are:

1. How can uncertainty quantification, specifically aleatoric and epistemic uncertainty, be better understood and applied in the context of active learning and active sampling?
2. How can a deeper understanding of the theoretical foundations of active learning and active sampling contribute to the progress of the field and improve the practical application of these techniques?

3. How can the cost of gathering and labeling data be reduced, and how can training be sped up in a principled fashion?
4. What are the connections between different active learning and active sampling approaches, and how can information theory be used to unify these approaches?

To answer these questions, we rely on *information theory* which helps quantify the information content of random variables and events, offers valuable insights and intuitions, and provides a principled framework for reasoning about uncertainty. Information diagrams (*I-diagrams*, see e.g. Figure 1.2) are particularly useful for building intuition, as they resemble Venn diagrams and stem from the realization that information-theoretic operations behave similarly to set operations [Yeung, 1991; Lang et al., 2022]. In a Bayesian setting, where model parameters are treated as random variables with associated uncertainty, information theory can be used to express and examine different concepts of 'informativeness' that are useful for data subset selection. This thesis examines objectives and extensions that follow from these intuitions and demonstrates that these objectives can be successfully applied in data subset selection.

In the next two sections of this chapter, we lay out the background and the structure of the thesis: §1.2 introduces necessary background material and points towards relevant literature for the thesis overall; and §1.3 provides an overview of the thesis and its structure.

1.2 Background & Literature Review

In this section, we introduce concepts that are relevant for the whole thesis. We revisit general background in information theory, Bayesian neural networks, active learning, and active sampling (in this order), and highlight important literature. Related work and background that are only relevant for their respective chapters are kept there: we hope that this aids the reader by keeping relevant information close to where it is needed. The next chapters generally follow the consistent notation introduced below, and we point out when we deviate from that in the specific chapters.

1.2.1 Information Theory

Information theory has provided insights for deep learning: information bottlenecks explain objectives both for supervised and unsupervised learning of high-dimensional data [Shwartz-Ziv and Tishby, 2017; Kirsch et al., 2020; Jónsson et al., 2020]; similarly, information theory has inspired Bayesian experiment design, Bayesian optimization, and active learning as well as motivated research into submodularity in general [Lindley, 1956; Foster et al., 2019].

A practical notation conveys valuable intuitions and concisely expresses new ideas. The currently employed notation in information theory, however, can be ambiguous for more complex expressions found in applied settings and often deviates between published works because researchers are from different backgrounds such as statistics, computer science, information engineering, which all use information theory. For example, $H(X, Y)$ is sometimes used to denote the *cross-entropy* between X and Y, which conflicts with common notation of the joint entropy $H(X, Y)$ for X and Y, or it is not clarified that $\mathrm{H}[X \mid Y]$ as conditional entropy of X given Y is an expectation over Y. Here, we present a disambiguated and consistent notation while striving to stay close to known notation when possible.

For a general introduction to information theory, we refer to Cover and Thomas [2005] and Yeung [2008]. In the following section, we introduce our practical and unified notation. In §2, we extend this notation to also include observed outcomes and point-wise mutual information, information gain and information-theoretic surprise. We start with notation that is explicit about the underlying probability distribution $\mathrm{p}(\,\cdot\,)$. Note that we do not require q to be normalized at this stage as it allows for greater notational flexibility.

Definition 1.1. Let Shannon's information content $\mathrm{H}(\,\cdot\,)$, cross-entropy $\mathrm{H}(\,\cdot\parallel\cdot\,)$, entropy $\mathrm{H}(\,\cdot\,)$, and KL divergence $\mathrm{D}_{\mathrm{KL}}(\,\cdot\parallel\cdot\,)$ (Kullback-Leibler divergence) be defined for a probability distribution p and non-negative function q for a random variable X and non-negative real number ρ as:

$$\mathrm{H}(\rho) \triangleq -\ln \rho \tag{1.1}$$
$$\mathrm{H}(\mathrm{p}(X) \parallel \mathrm{q}(X)) \triangleq \mathbb{E}_{\mathrm{p}(x)}\, \mathrm{H}(\mathrm{q}(x)) \tag{1.2}$$
$$\mathrm{H}(\mathrm{p}(X)) \triangleq \mathrm{H}(\mathrm{p}(X) \parallel \mathrm{p}(X)) \tag{1.3}$$
$$\mathrm{D}_{\mathrm{KL}}(\mathrm{p}(X) \parallel \mathrm{q}(X)) \triangleq \mathrm{H}(\mathrm{p}(X) \parallel \mathrm{q}(X)) - \mathrm{H}(\mathrm{p}(X)), \tag{1.4}$$

where we fix $\mathrm{H}(0) \triangleq -\infty$ with $0 \cdot \mathrm{H}(0) = 0$ as usual.

Shannon [1948] introduced the information content as negative logarithm due to its additivity for independent messages: $\mathrm{H}(\mathrm{p}(x, y)) = \mathrm{H}(\mathrm{p}(x)) + \mathrm{H}(\mathrm{p}(y))$ for independent random variables X and Y.

Proposition 1.1. *For a random variable X with probability distributions* p, p_1 *and* p_2*, and non-negative functions* q, q_1 *and* q_2 *and $\alpha \in [0, 1]$:*

$$\mathrm{H}(\mathrm{p} \parallel \alpha\, \mathrm{q}) = \mathrm{H}(\mathrm{p} \parallel \mathrm{q}) + \mathrm{H}(\alpha), \tag{1.5}$$
$$\mathrm{H}(\mathrm{p} \parallel \mathrm{q}^{\alpha}) = \alpha\, \mathrm{H}(\mathrm{p} \parallel \mathrm{q}) \tag{1.6}$$
$$\mathrm{H}(\mathrm{p} \parallel \mathrm{q}_1\, \mathrm{q}_2) = \mathrm{H}(\mathrm{p} \parallel \mathrm{q}_1) + \mathrm{H}(\mathrm{p} \parallel \mathrm{q}_2), \tag{1.7}$$
$$\mathrm{H}(\alpha\, \mathrm{p}_1 + (1-\alpha)\, \mathrm{p}_2 \parallel \mathrm{q}) = \tag{1.8}$$
$$= \alpha\, \mathrm{H}(\mathrm{p}_1 \parallel \mathrm{q}) + (1-\alpha)\, \mathrm{H}(\mathrm{p}_2 \parallel \mathrm{q}) \tag{1.9}$$
$$= \mathrm{H}(\mathrm{p}_1 \parallel \mathrm{q}^{\alpha}) + \mathrm{H}(\mathrm{p}_2 \parallel \mathrm{q}^{1-\alpha}), \tag{1.10}$$

where we have left out "(X)" everywhere for brevity.

Proof. The statements follow from the linearity of the expectation and the additivity of the logarithm for products. □

This can be extended to show that cross-entropies are linear in their left-hand argument and log-linear in their right-hand argument.

When we want to emphasize that we approximate the true distribution p using a different distribution q and the true probability distribution p is understood, we use the notation $\mathrm{H}_{\mathrm{q}}[\,\cdot\,]$ for $\mathrm{H}(\mathrm{p}(\,\cdot\,) \parallel \mathrm{q}(\,\cdot\,))$ following notation in Kirsch et al. [2020] and Xu et al. [2020]:

Definition 1.2. When the true probability distribution p is understood from context, we will use the following shorthand notation:

$$\mathrm{H}[X] \triangleq \mathrm{H}(\mathrm{p}(X)) \tag{1.11}$$

$$\mathrm{H_q}[X] \triangleq \mathrm{H}(\mathrm{p}(X) \parallel \mathrm{q}(X)). \tag{1.12}$$

When we have a parameterized distribution q_θ with parameters θ, we will write $\mathrm{H}_\theta[\,\cdot\,]$ instead of $\mathrm{H}_{\mathrm{q}_\theta}[\,\cdot\,]$ when the context is clear.

Approximating a possibly intractable distribution with a parameterized one is common when performing variational inference. The main motivation for this notation is that when q is a density, $\int q(x)\,dx = 1$, we have $\mathrm{H_q}[\,\cdot\,] \geq \mathrm{H}[\,\cdot\,]$ with equality when $\mathrm{q} = \mathrm{p}$. Concretely, we have the following useful identities:

Proposition 1.2. *We have the following lower-bounds for the cross-entropy and KL, with* $Z_q \triangleq \int q(x)\,dx$*:*

$$\mathrm{H}(\mathrm{p}(X) \parallel \mathrm{q}(X)) \geq \mathrm{H}(\mathrm{p}(X)) + \mathrm{H}(Z_q), \tag{1.13}$$
$$\mathrm{D_{KL}}(\mathrm{p}(X) \parallel \mathrm{q}(X)) \geq \mathrm{H}(Z_q), \tag{1.14}$$

with equality exactly when $\mathrm{q}/Z_q = \mathrm{p}$ *for* $Z_q \triangleq \int q(x)\,dx$*.*

Proof. The statements follow from Jensen's inequality and the convexity of $\mathrm{H}(\,\cdot\,)$. □

This also implies the non-negativity of the KL for densities when we substitute $Z_q = 1$ in above statements. We repeat the result as it is often used:

Corollary 1.3. *When q is a probability distribution, we have:*

$$\mathrm{H}(\mathrm{p}(X) \parallel \mathrm{q}(X)) \geq \mathrm{H}(\mathrm{p}(X)), \tag{1.15}$$
$$\mathrm{D_{KL}}(\mathrm{p}(X) \parallel \mathrm{q}(X)) \geq 0, \tag{1.16}$$

with equality exactly when $\mathrm{q} = \mathrm{p}$.

Note that for continuous distributions, above equality $\mathrm{p} = \mathrm{q}$ only has to hold almost everywhere.

Above definitions are trivially extended to joints of random variables by substituting the random variable of the product space. Similarly, the conditional entropy is defined by taking the expectation over both X and Y. For example:

Proposition 1.4. *Given random variables* X *and* Y*, we have:*

$$\mathrm{H}[X, Y] = \mathbb{E}_{\mathrm{p}(x,y)}\, \mathrm{H}(\mathrm{p}(x, y)); \tag{1.17}$$
$$\mathrm{H}[X \mid Y] = \mathbb{E}_{\mathrm{p}(x,y)}\, \mathrm{H}(\mathrm{p}(x \mid y)). \tag{1.18}$$

In particular, note that $\mathrm{H}[X \mid Y]$ is an expectation over X *and* Y.

For cross-entropies and KL divergences, we expand the definitions similarly. In particular, we have the following equality for cross-entropies, which follows from these definitions:

$$\mathrm{H}(\mathrm{p}(X \mid Y) \parallel \mathrm{q}(X \mid Y)) = \mathbb{E}_{\mathrm{p}(x,y)}\, \mathrm{H}(\mathrm{q}(x \mid y)) \quad = \mathrm{H}(\mathrm{p}(X, Y) \parallel \mathrm{q}(X \mid Y)). \tag{1.19}$$

The last idiosyncrasy only applies to cross-entropies. For KL divergences we have:

$$\mathrm{D_{KL}}(\mathrm{p}(X \mid Y) \parallel \mathrm{q}(X \mid Y)) = \mathrm{H}(\mathrm{p}(X, Y) \parallel \mathrm{q}(X \mid Y)) - \mathrm{H}(\mathrm{p}(X \mid Y)) \tag{1.20}$$

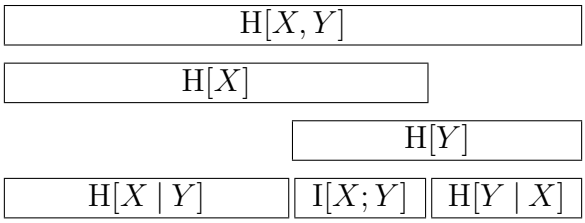


Figure 1.1: *Reproduction of Figure 8.1 from MacKay [2003] using the new suggested notation:* The relationship between joint entropy, marginal entropy, conditional entropy and mutual information.

$$\mathrm{D_{KL}}(\mathrm{p}(X,Y) \parallel \mathrm{q}(X \mid Y)) = \mathrm{H}(\mathrm{p}(X,Y) \parallel \mathrm{q}(X \mid Y)) - \mathrm{H}(\mathrm{p}(X,Y)). \tag{1.21}$$

The second terms on each right-hand side are usually not equal $\mathrm{H}(\mathrm{p}(X \mid Y)) \neq \mathrm{H}(\mathrm{p}(X,Y))$, and the two expressions are thus not equivalent. The reader might wonder when we are interested in $\mathrm{D_{KL}}(\mathrm{p}(X,Y) \parallel \mathrm{q}(X \mid Y))$. It can arise when performing symbolic manipulations, so we mention it explicitly here.

The mutual information and point-wise mutual information [Fano, 1962; Church and Hanks, 1989] are defined as:

Definition 1.3. For random variables X and Y and outcomes x and y respectively, the point-wise mutual information $\mathrm{I}[x;y]$ and the mutual information $\mathrm{I}[X;Y]$ are:

$$\mathrm{I}[x;y] \triangleq \mathrm{H}[x] - \mathrm{H}[x \mid y] = \mathrm{H}(\frac{\mathrm{p}(x)\,\mathrm{p}(y)}{\mathrm{p}(x,y)}) \tag{1.22}$$

$$\mathrm{I}[X;Y] \triangleq \mathrm{H}[X] - \mathrm{H}[X \mid Y] = \mathbb{E}_{\mathrm{p}(x,y)}\,\mathrm{I}[x;y]. \tag{1.23}$$

This is similarly extended to $\mathrm{I}[X;Y \mid Z] = \mathrm{H}[X \mid Z] - \mathrm{H}[X \mid Y,Z]$ or $\mathrm{I}[X_1,X_2;Y] = \mathrm{H}[X_1,X_2] - \mathrm{H}[X_1,X_2 \mid Y]$ and so on.

MacKay [2003] has an elegant visualization for information quantities, which we reproduce in Figure 1.1.

1.2.1.1 Information Diagrams (I-Diagrams)

Similarly, Yeung [1991] introduces *information diagrams (I-diagrams)* which provide another useful intuitive approach: they show that the intuitions that map information quantities to set expressions can be made principled using a specially defined signed information measure. Note that interaction information [McGill, 1954] follows as canonical generalization of the mutual information to multiple variables from that work. Lang et al. [2022] further generalize these results.

Information diagrams, like the one depicted in Figure 1.2, visualize the relationship between information quantities: Yeung [1991] shows that we can define a signed measure μ^* such that these well-known quantities map to abstract sets and are consistent with set operations.

$$\mu^*[A] = \mathrm{H}[A]$$
$$\mu^*[\cup_i A_i] = \mathrm{H}[A_1,\ldots,A_n]$$
$$\mu^*[\cup_i A_i - \cup_i B_i] = \mathrm{H}[A_1,\ldots,A_n \mid B_1,\ldots,B_n]$$

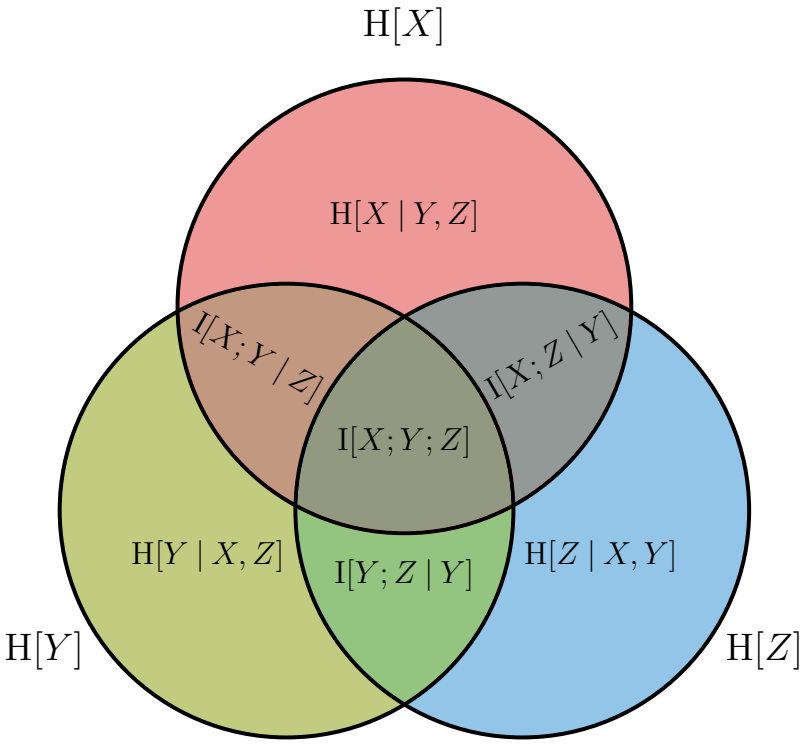


Figure 1.2: Example of an I-Diagram for three random variables X, Y, Z. All seven atomic quantities are depicted as well as the overall entropies $\mathrm{H}[X], \mathrm{H}[Y], \mathrm{H}[Z]$.

$$\mu^*[\cap_i A_i] = \mathrm{I}[A_1; \ldots; A_n]$$
$$\mu^*[\cap_i A_i - \cup_i B_i] = \mathrm{I}[A_1; \ldots; A_n \mid B_1, \ldots, B_n]$$

In other words, equalities can be read off directly from I-diagrams: an information quantity is the sum of its parts in the corresponding I-diagram. This is similar to Venn diagrams. The sets used in I-diagrams are just abstract symbolic objects, however.

An important distinction between I-diagrams and Venn diagrams is that while we can always read off inequalities in Venn diagrams, this is not true for I-diagrams in general because mutual information terms in more than two variables can be negative. In Venn diagrams, a set is always larger or equal any subset.

However, if we show that all information quantities are non-negative, we can treat an I-diagram like a Venn diagram and read off both equalities and inequalities from it. Nevertheless, caution is warranted sometimes. As the signed measure can be negative, $\mu^*[X \cap Y] = 0$ does *not* imply $X \cap Y = \emptyset$: deducing that a mutual information term is 0 does not imply that one can simply remove the corresponding area in the I-diagram. There could be Z with $\mu^*[(X \cap Y) \cap Z] < 0$, such that $\mu^*[X \cap Y] = \mu^*[X \cap Y \cap Z] + \mu^*[X \cap Y - Z] = 0$ but $X \cap Y \neq \emptyset$. This also means that we cannot drop the term from expressions when performing symbolic manipulation, and we cannot remove the area from an I-diagram without loss of generality:

While a mutual information term of two random variables equalling zero implies those two random variables are independent, the mutual information of those two random variables with a third might not be zero, and thus does not allow us to draw them as disjoint areas.

The only time when one can safely remove an area from the diagram is for *atomic* quantities, which are quantities which reference all the available random variables [Yeung, 1991]. For example, when we only have three variables X, Y, Z, $\mathrm{I}[X;Y;Z]$ and $\mathrm{I}[X;Y \mid Z]$ are atomic quantities, like in Figure 1.2. We can safely remove atomic quantities from I-diagrams when they are 0 because there are no other random variables left that could lead to the issues described above.

Continuing the example, for $\mathrm{I}[X;Y] = \mu^*[X \cap Y] = 0$, having $0 = \mathrm{I}[X;Y;Z] = \mu^*[X \cap Y \cap Z]$ would imply $X \cap Y \cap Z = \emptyset$, and we could remove that area from the diagram without loss of generality. Moreover, the atomic quantity $\mathrm{I}[X;Y \mid Z] = \mu^*[X \cap Y - Z] = 0$ and could be removed from the diagram as well in that case.

We only use I-diagrams for the three variable case, but they supply us with tools to easily come up with equalities and inequalities for information quantities. In the general case with multiple variables, they can be difficult to draw, but for Markov chains they can be of great use [Yeung, 1991].

1.2.2 Bayesian Neural Networks (BNNs)

In this thesis, we focus on Bayesian neural networks (BNNs) [Neal, 1995; MacKay, 1992c]. Unlike regular neural networks, BNNs maintain a distribution over their weights instead of point estimates. This allows for better uncertainty quantification and for disentangling different types of uncertainties [Kendall and Gal, 2017]. Compared to other Bayesian approaches, BNNs scale well to high-dimensional inputs, such as images while remaining close to neural networks conceptually, allowing to apply advances in deep learning in Bayesian settings [Sharma et al., 2022; Gal et al., 2017]. Hence, BNNs have become a powerful alternative to traditional neural networks.

Challenges for BNNs. However, performing exact inference in BNNs is intractable for any reasonably sized model, so we resort to using a variational approximation. The intractability of exact Bayesian inference in deep learning has led to the development of approximate inference methods [Hinton and van Camp, 1993; Hernández-Lobato and Adams, 2015; Blundell et al., 2015; Gal and Ghahramani, 2016a]. Improvements in approximate inference [Blundell et al., 2015; Gal and Ghahramani, 2016a] have enabled their usage for high dimensional data such as image for Bayesian active learning of images [Gal et al., 2017]. Similar to Gal et al. [2017], we will mainly use MC dropout [Gal and Ghahramani, 2016a], which is easy to implement, scales well to large models and datasets, and is straightforward to optimize. Deep Ensembles [Lakshminarayanan et al., 2017] can also be considered as an alternative to BNNs.

Probabilistic Model. We consider supervised learning of a probabilistic predictive model, $\mathrm{p}(y, \boldsymbol{\omega} \mid \boldsymbol{x})$, where X is an input, Y is a label, and $\boldsymbol{\Omega}$ are the model parameters of our model $\mathcal{M}$ with prior distribution $\mathrm{p}(\boldsymbol{\omega})$:

$$\mathrm{p}(y, \boldsymbol{\omega} \mid \boldsymbol{x}) = \mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})\, \mathrm{p}(\boldsymbol{\omega}). \tag{1.24}$$

Importantly, we assume that the model's predictions are independent given $\boldsymbol{\Omega}$, such that we can write:

$$\mathrm{p}(y_1, \ldots, y_n \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{\omega}) = \prod_{i=1}^{n} \mathrm{p}(y_i \mid \boldsymbol{x}_i, \boldsymbol{\omega}). \tag{1.25}$$

We assume classification tasks, and thus $Y \in [\mathtt{C}] \triangleq \{1, \ldots, \mathtt{C}\}$. The exceptions are §10 and Appendix A.

Datasets. Further, we assume we have datasets $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, with $\mathcal{D}^{\text{test}} \sim \hat{\text{p}}_{\text{true}}(y, \boldsymbol{x})$, the 'true' underlying data distribution. We use $\mathcal{D}$ to represent additional data that we might condition on. We will define additional datasets in the respective chapters and as we go along (for example, the unlabeled pool set in active learning as $\mathcal{D}^{\text{pool}}$).

Bayesian Model Averaging and Bayesian Inference. We are interested in the *BMA (Bayesian model averaging)* prediction given the Bayesian posterior $\text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$. Bayesian model averaging (BMA) is performed by marginalizing over $\boldsymbol{\Omega}$ to obtain the *predictive distribution* $\text{p}(y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}})$:

$$\text{p}(y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}) = \mathbb{E}_{\text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})}[\text{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})], \tag{1.26}$$

where we use *Bayesian inference* to obtain a posterior $\text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$ for data $\mathcal{D}^{\text{train}}$ by:

$$\text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}}) \propto \text{p}(\mathcal{D}^{\text{train}} \mid \boldsymbol{\omega})\, \text{p}(\boldsymbol{\omega}). \tag{1.27}$$

$\text{p}(\mathcal{D}^{\text{train}} \mid \boldsymbol{\omega})$ is the likelihood of the data given the parameters $\boldsymbol{\Omega}$, and $\text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$ is the new posterior distribution over $\boldsymbol{\Omega}$.

Variational Inference. As mentioned, Bayesian inference is often intractable, and instead, we use variational inference methods to approximate the posterior $\text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$ via a simpler distribution $\text{q}(\boldsymbol{\omega}) \approx \text{p}(\boldsymbol{\omega} | \mathcal{D}^{\text{train}})$. Usually, this is phrased as an optimization problem where we try to find the distribution q from a variational family of potential distributions that minimizes a given divergence measure between $\text{q}(\boldsymbol{\omega})$ and $\text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$:

$$\text{q}(\boldsymbol{\omega}) = \arg\min_{\text{q}(\boldsymbol{\omega})} \text{D}(\text{q}(\boldsymbol{\omega}) \,||\, \text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})). \tag{1.28}$$

We will use the KL divergence $\text{D}_{\text{KL}}(\text{q}(\boldsymbol{\omega}) \parallel \text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}}))$ as divergence measure in this thesis, but other divergence measures have been proposed and are used in practice, as well. Instead of above KL divergence which is usually intractable, the *ELBO (Evidence Lower Bound)* is commonly used to optimize the variational approximation instead. By expanding above KL divergence, we obtain:

$$\begin{aligned} &\text{D}_{\text{KL}}(\text{q}(\boldsymbol{\omega}) \parallel \text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})) && (1.29)\\ &= \mathbb{E}_{\text{q}(\boldsymbol{\omega})}[\underbrace{-\log \text{p}(y^{\text{train}}_{1..\text{N}} \mid \boldsymbol{x}^{\text{train}}_{1..\text{N}}, \boldsymbol{\omega})}_{\text{log likelihood}}] + \underbrace{\text{D}_{\text{KL}}(\text{q}(\boldsymbol{\omega}) \parallel \text{p}(\boldsymbol{\omega}))}_{\text{prior regularization}} + \underbrace{\log \text{p}(\mathcal{D}^{\text{train}})}_{\text{model evidence}} \geq 0, && (1.30) \end{aligned}$$

The model evidence is independent of $\text{q}(\boldsymbol{\omega})$. It can be ignored during optimization. Rearranging, leaves us with the ELBO, which we then maximize:

$$\mathbb{E}_{\text{q}(\boldsymbol{\omega})}[\log \text{p}(y^{\text{train}}_{1..\text{N}} \mid \boldsymbol{x}^{\text{train}}_{1..\text{N}}, \boldsymbol{\omega})] - \text{D}_{\text{KL}}(\text{q}(\boldsymbol{\omega}) \parallel \text{p}(\boldsymbol{\omega})) \leq \log \text{p}(\mathcal{D}^{\text{train}}). \tag{1.31}$$

We can use the local reparameterization trick and Monte-Carlo (MC) dropout to obtain an implicit $\text{q}(\boldsymbol{\omega})$ distribution we can draw samples from in a deep learning context [Gal and Ghahramani, 2016a]. The posterior BMA using likelihood $\text{p}(\mathcal{D}^{\text{train}} \mid \boldsymbol{\omega})$ and prior $\text{p}(\boldsymbol{\omega})$ is then approximated by:

$$\text{p}(y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}) \approx \mathbb{E}_{\text{q}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})}[\text{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})]. \tag{1.32}$$

We often omit making this last step explicit in our deductions and use $\text{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$ when, in practice, we would substitute $\text{q}(\boldsymbol{\omega})$.

Stochastic Parameters and Model Choices. Importantly, we often use of Bayesian models in the narrow sense that we only require some *stochastic* parameters $\boldsymbol{\Omega}$ with a distribution $p(\boldsymbol{\omega})$. This choice of $p(\boldsymbol{\omega})$ covers (deep) ensembles [Dietterich, 2000; Lakshminarayanan et al., 2017], neural networks with stochasticity in a subset of parameters [Sharma et al., 2022], as well as models with additional stochastic inputs [Osband et al., 2021a] or randomized training data through subsampling of the training set, e.g., bagging [Breiman, 1996]. Similar views have been put forward by He et al. [2020]; Wilson and Izmailov [2020] for deep ensembles, and for random forests [Shaker and Hüllermeier, 2020].

1.2.3 Uncertainty Quantification

Uncertainty quantification is a broad field, but two types of uncertainty are often of interest in machine learning: *epistemic uncertainty*, which is inherent to the model, caused by a lack of training data, and hence reducible with more data, and *aleatoric uncertainty*, caused by inherent noise or ambiguity in data, and hence irreducible with more data [Der Kiureghian and Ditlevsen, 2009]. Disentangling these two is critical for tasks such as:

- **Active Learning.** In active learning, we want to avoid inputs with high aleatoric and low epistemic uncertainty, as these will not be informative for the model [Gal et al., 2017].
- **Out-of-Distribution Detection.** In OoD detection [Hendrycks and Gimpel, 2017], we want to avoid flagging ambiguous in-distribution (iD) examples as OoD.
- **Deferral of Uncertain Predictions.** To defer predictions [Filos et al., 2019], we seek inputs with either high epistemic uncertainty or high aleatoric uncertainty, for different purposes: in the former case, we want to defer to an expert for a decision, while in the latter case, we want to defer to the data source for a better measurement.

These tasks and distinctions matter in particular for noisy and ambiguous datasets found in safety-critical applications like autonomous driving [Huang and Chen, 2020] and medical diagnosis [Esteva et al., 2017; Filos et al., 2019].

While there are many ways to approximate these uncertainties using metrics, we will focus on three in the next chapters: *mutual information*, *entropy*, and *feature-space density*.

In this subsection, we will define epistemic and aleatoric uncertainty in more detail and give a brief overview of these three metrics and relevant concepts:

Epistemic Uncertainty. For point $\boldsymbol{x}$, epistemic uncertainty is a quantity which is high for a previously unseen $\boldsymbol{x}$, and decreases when $\boldsymbol{x}$ is added to the training set and the model is updated [Kendall and Gal, 2017]. This conforms with using mutual information in Bayesian models and deep ensembles [Smith and Gal, 2018] and feature-space density in deterministic models as surrogates for epistemic uncertainty [Postels et al., 2020] as we will examine in chapter §3.

Aleatoric Uncertainty. For a point $\boldsymbol{x}$, aleatoric uncertainty is a quantity which is high for ambiguous or noisy samples [Kendall and Gal, 2017]. For example, in classification, aleatoric uncertainty will be high when multiple labels were to be observed at $\boldsymbol{x}$. Crucially, aleatoric uncertainty does not decrease with more data because it is inherent to the data. As we gather more data, it can actually increase as

we might have undersampled the data distribution. Note that aleatoric uncertainty is only meaningful in-distribution, as, by definition, it quantifies the level of noise or ambiguity which might be observed. More practically speaking, if the probability of observing $\boldsymbol{x}$ under the data generating distribution is zero, the conditional probability $\mathrm{p}(y \mid \boldsymbol{x}) = \frac{\mathrm{p}(\boldsymbol{x}, y)}{\mathrm{p}(\boldsymbol{x})}$ is undefined, and hence, neither is the respective entropy as a measure of aleatoric uncertainty.

Bayesian Models. To measure uncertainty, principled approaches exist for Bayesian models [Gal, 2016]. Given a Bayesian model $\mathrm{p}(y, \boldsymbol{\omega} \mid \boldsymbol{x})$, its predictive entropy $\mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ upper-bounds the epistemic uncertainty, where epistemic uncertainty is quantified as the mutual information $\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ between parameters $\boldsymbol{\Omega}$ and output Y for a given input $\boldsymbol{x}$ [Gal, 2016; Smith and Gal, 2018]:

$$\underbrace{\mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]}_{\text{predictive}} = \underbrace{\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]}_{\text{epistemic}} + \underbrace{\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}]}_{\text{aleatoric (for iD } \boldsymbol{x})} \tag{1.33}$$

$$\Leftrightarrow \mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] = \mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}]. \tag{1.34}$$

Predictive entropy will be high for both iD ambiguous samples (high aleatoric uncertainty) and for OoD samples (high epistemic uncertainty). Hence, predictive entropy is a good measure for informativeness for active learning or as a metric for OoD detection only when used with curated datasets that do not contain ambiguous samples. Note that aleatoric uncertainty is only meaningful in-distribution because it quantifies the level of noise or ambiguity which might be observed for input $\boldsymbol{x}$.

Looking at the two terms in equation (1.34), for the mutual information to be high, the left term has to be high and the right term low. The left term is the entropy of the model prediction, which is high when the model's prediction is uncertain. The right term is an expectation of the entropy of the model prediction over the posterior of the model parameters and is low when the model is overall certain for each draw of model parameters from the posterior. Both can only happen when the model has many possible ways to explain the data, which means that the different models induced by different parameter samples are disagreeing among themselves.

This intuitively satisfies the definition of epistemic uncertainty above, as adding a point to the training set ought to decrease the epistemic uncertainty as there will be less disagreement among the models induced by different parameter samples after training with the point.

Deep Ensembles. The predictions of a deep ensemble [Lakshminarayanan et al., 2017] are the average of the outputs of an ensemble of neural networks. The total uncertainty of the prediction is then estimated as the entropy of the averaged softmax outputs. This can be viewed as approximating the BMA over the distribution of all possibly trained models [Wilson and Izmailov, 2020], as each ensemble member, producing a softmax output $\mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})$, can be considered to be drawn from some distribution $\mathrm{p}(\boldsymbol{\omega})$ of the possibly trained model parameters $\boldsymbol{\Omega}$, which is induced by the push forward of the weight initialization under stochastic optimization. As a result, Equation 1.34 can also be applied to Deep Ensembles to disentangle aleatoric and epistemic uncertainty from predictive uncertainty.

Despite the high computational overhead at training and test time, Deep Ensembles along with recent extensions [Smith and Gal, 2018; Wen et al., 2020; Dusenberry et al., 2020] form the state-of-the-art in uncertainty quantification in deep learning.

In practice, both mutual information $\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ and predictive entropy $\mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ are used in the literature for active learning and to detect OoD data, but predictive entropy will be high whenever either epistemic uncertainty is high, or when aleatoric uncertainty is high: it upper-bounds the mutual information. This can help separate iD and OoD data better for curated iD datasets, offering an explanation for previous empirical findings of predictive entropy outperforming mutual information [Malinin and Gales, 2018]. With ambiguous iD samples, it can lead to confounding, however, which we analyze in chapter §3.

Deterministic Models. A single deterministic model, in the sense that we use a single parameter point estimate, produces a softmax distribution $\mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})$, and commonly either the *softmax confidence* $\max_c \mathrm{p}(y = c \mid \boldsymbol{x}, \boldsymbol{\omega})$ or the *softmax entropy* $\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}]$ are used as a measure of uncertainty [Hendrycks and Gimpel, 2017]. In active learning, the softmax entropy and confidence are often used as a baseline measure of informativeness. Note that the confidence is known as variation ratios in active learning Freeman [1965]. In both settings, they do not perform as well as deep ensembles [Beluch et al., 2018].

Popular approaches to improve these metrics include pre-processing of inputs and post-hoc calibration methods [Liang et al., 2018; Guo et al., 2017], alternative objective functions [Lee et al., 2018a; DeVries and Taylor, 2018], and exposure to outliers [Hendrycks et al., 2019]. However, these methods are known to suffer from shortcomings: they can fail under distribution shift [Ovadia et al., 2019], require significant changes to the training setup, or assume the availability of OoD samples during training.

Feature-Space Distances & Feature-Space Density. Feature-space distances [Lee et al., 2018b; van Amersfoort et al., 2020; Liu et al., 2020a] and feature-space density [Postels et al., 2020; Settles, 2010] based on the training set offer a different approach for estimating uncertainty in deterministic models: satisfying the definition of epistemic uncertainty above, they decrease when previously unseen samples are added to the training set. This is the case for feature-space distance and density methods because they estimate distance or density, respectively, of the training data in feature space. A previously unseen point with high distance (low density), once added to the training data, will have low distance (high density). Hence, they can be used as a proxy for epistemic uncertainty, under important assumptions about the feature space as detailed below. We will examine this in more detail in §3. None of these methods, however, is competitive with Deep Ensembles, in uncertainty quantification, potentially for the reasons discussed next.

Feature Collapse. van Amersfoort et al. [2020] argue that feature collapse is why distance and density estimation in the feature space may fail to capture epistemic uncertainty: feature extractors might map the features of OoD inputs into iD regions in the feature space [van Amersfoort et al., 2021], making it difficult for the later stages to distinguish between iD and OoD samples.

Smoothness & Sensitivity. To prevent feature collapse, smoothness and sensitivity can be encouraged by subjecting the feature extractor f_θ, with parameters θ to a *bi-Lipschitz constraint*:

$$K_L \, d_I(\boldsymbol{x}_1, \boldsymbol{x}_2) \leq d_F(f_\theta(\boldsymbol{x}_1), f_\theta(\boldsymbol{x}_2)) \leq K_U \, d_I(\boldsymbol{x}_1, \boldsymbol{x}_2),$$

for all inputs, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, where d_I and d_F denote metrics for the input and feature space respectively, and K_L and K_U the lower and upper Lipschitz constants [Liu et al., 2020a]. The lower bound ensures *sensitivity* to distances in the input space, and the upper bound ensures *smoothness* in the features, preventing them from becoming too sensitive to input variations, which, otherwise, can lead to poor generalization and loss of robustness [van Amersfoort et al., 2020]. Methods of encouraging bi-Lipschitzness include: **i)** gradient penalty, by applying a two-sided penalty to the L2 norm of the Jacobian [Gulrajani et al., 2017], and **ii)** spectral normalization [Miyato et al., 2018] in models with residual connections, like ResNets [He et al., 2016]. [Smith et al., 2021] provides an in-depth analysis which supports that spectral normalization leads to bi-Lipschitzness. Compared to the Jacobian gradient penalty [van Amersfoort et al., 2020], spectral normalization is significantly faster and has more stable training dynamics.

1.2.4 Active Learning

One key problem in deep learning is data efficiency. While excellent performance can be obtained with modern approaches, these are often data-hungry, rendering the deployment of deep learning in the real-world challenging for many tasks. Active learning [Atlas et al., 1989; Cohn et al., 1996] is a powerful technique for improving labelling efficiency; it has a rich history in the machine learning community, with its origins dating back to seminal works such as Atlas et al. [1989]; Lindley [1956]; Fedorov [1972]; MacKay [1992b]. A comprehensive survey of early active learning methods can be found in Settles [2010], while more recent surveys of contemporary deep learning methods can be found in Ren et al. [2022] and Zhan et al. [2022a].

Instead of a priori collecting and labelling a large dataset, which often comes at a significant expense, active learning provides a mechanism for effective training of machine learning models in settings where unlabeled data is plentiful, but labelling is expensive by carefully selecting which data points to acquire labels for, using information from previously acquired data to establish the points whose labels will be most informative for training. After each acquisition step, the newly labeled points are added to the training set, and the model is retrained. This process is repeated until a suitable level of accuracy is achieved. The goal of active learning is to minimize the amount of data that needs to be labeled. Active learning has made real-world impact in manufacturing [Tong, 2001], robotics [Calinon et al., 2007], recommender systems [Adomavicius and Tuzhilin, 2005], medical imaging [Hoi et al., 2006], and NLP [Siddhant and Lipton, 2018], motivating further exploration of this fascinating topic.

Origins. The conceptual origins of active learning can be traced back to Bayesian-optimal experiment design [Chaloner and Verdinelli, 1995a; Lindley, 1956; Rainforth et al., 2023]. In machine learning, the non-Bayesian approaches started in the sequential (stream-based) setting as *selective sampling* [Atlas et al., 1989] before being referred to more generally as *active learning* [Atlas et al., 1989]. Bayesian methods and active

learning were connected early on by e.g. MacKay [1992b] with objectives that are still highly relevant today—as we will see, these objectives are especially relevant for this thesis. An excellent literature review of the original non-Bayesian active learning paradigms can be found in Settles [2010].

Bayesian Optimization. Active learning is also closely related to *Bayesian Optimization (BO)*, which is a well-established methodology for global optimization of black-box functions [Mockus, 1974; Jones et al., 1998], especially when the function evaluations are expensive. Typical applications of BO span from machine learning and statistics to engineering and experimental design [Snoek et al., 2012; Shahriari et al., 2016; Saleh et al., 2022] under various constraints [Nguyen et al., 2017; Siivola et al., 2021].

BO is based on building a probabilistic surrogate model of the black-box function, often a Gaussian process [Williams and Rasmussen, 2006], and using an acquisition function to drive the search towards the regions of the input space that are likely to improve upon the current best solution [Shahriari et al., 2016]. The queries are sequentially selected and have to balance exploration and exploitation [Srinivas et al., 2010].

Many BO methods utilize the expected improvement heuristic [Hernández-Lobato et al., 2014, 2015, 2016] or rely on Thompson sampling (TS) [Thompson, 1933; Russo and Roy, 2013].

Batch Bayesian Optimization. The classical BO setting is sequential, where the next query directly depends on the outcomes of the previous evaluations. However, this does not take into account that often experiments can be conducted in parallel, such as in large-scale computing environments or in laboratory experiments where measurements may come from different sources and may introduce significant waiting times otherwise [Folch et al., 2023].

To address this, *batch Bayesian optimization (BBO)* methods have been developed, where multiple decisions are made simultaneously and evaluated in parallel, providing a significant acceleration over the sequential setting [Groves and Pyzer-Knapp, 2018; Chowdhury and Gopalan, 2019; Oh et al., 2021]. Many BBO methods have been proposed in the literature, including methods based on the expected improvement heuristic [Shah and Ghahramani, 2015; González et al., 2016; Alvi et al., 2019], Thompson sampling [Kandasamy et al., 2018], and upper confidence bound [Contal et al., 2013; Daxberger and Low, 2017].

Semi-Supervised Learning. A related approach to active learning is semi-supervised learning (also sometimes referred to as weakly-supervised), in which the labeled data is commonly assumed to be fixed, and the unlabeled data is used for unsupervised learning [Kingma et al., 2014; Rasmus et al., 2015]. Wang et al. [2017]; Sener and Savarese [2018]; Sinha et al. [2019] explore combining it with active learning.

Pool-Based Active Learning. Active learning is most often done in a *pool*-based setting [Lewis and Gale, 1994], wherein we start with a large reservoir of unlabeled data points, known as the *pool set* $\mathcal{D}^{\text{pool}}$, from which we sequentially choose points to label, after which they are removed from $\mathcal{D}^{\text{pool}}$ and added to the *training dataset* $\mathcal{D}^{\text{train}}$ together with their acquired label. In this thesis, we focus on such settings. The steps of an active learning loop in a pool-based setting are depicted in Figure 1.3(a). The main challenge in pool-based batch active learning is the choice of the acquisition function.

Acquisition Functions. The mechanism by which we choose points to label is known as an acquisition strategy and most commonly corresponds to choosing the data point which maximizes a prespecified *acquisition function* that reflects the utility of acquiring a label for that point. (Similarly, as we will see, when we perform batch acquisition, we will want to find the batch of points that maximize a joint utility score.) We will define acquisition functions as functions that return a score for an individual sample $a(\boldsymbol{x}, \mathcal{M})$ given the current model $\mathcal{M}$. We then acquire labels for the sample that maximizes the score:

$$\boldsymbol{x}^{\text{acq},*} \triangleq \underset{\boldsymbol{x}^{\text{acq}} \in \mathcal{D}^{\text{pool}}}{\arg\max}\, a(\boldsymbol{x}^{\text{acq}}, \mathcal{M}). \tag{1.35}$$

There are several simple acquisition functions which are often used as baselines [Gal et al., 2017]:

Entropy. The predictive entropy is used as acquisition score. It is defined as:

$$a_{\text{Entropy}}(\boldsymbol{x}; \mathcal{M}) \triangleq \mathrm{H}[Y \mid \boldsymbol{x}]. \tag{1.36}$$

It is non-negative and measures the total uncertainty that model assigns to an input $\boldsymbol{x}$.

Variation Ratio. Also known as "*least confidence*", the variation-ratio is the complement of the most-confident class prediction and thus selects samples with the lowest confidence for acquisition:

$$a_{\text{Variation-Ratios}}(\boldsymbol{x}; \mathcal{M}) \triangleq 1 - \max_{y} \mathrm{p}(y \mid \boldsymbol{x}). \tag{1.37}$$

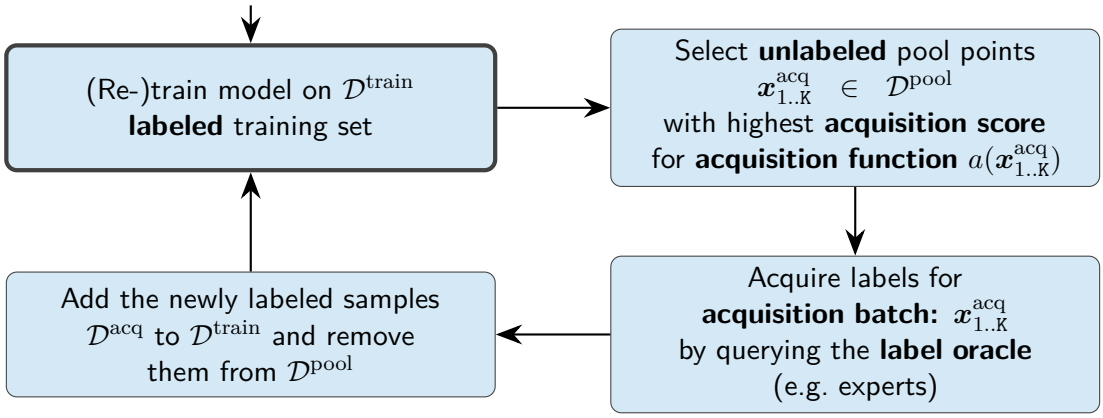
This scoring function is non-negative and a score of 0 means that the sample is uninformative: a score of 0 means that the respective prediction is one-hot, and then that the expected information gain is also 0, as can be easily verified.

Standard Deviation. The standard deviation score function measures the sum of the class probability deviations and is closely related to the BALD scores (Proposition B.9, Smith and Gal [2018]):

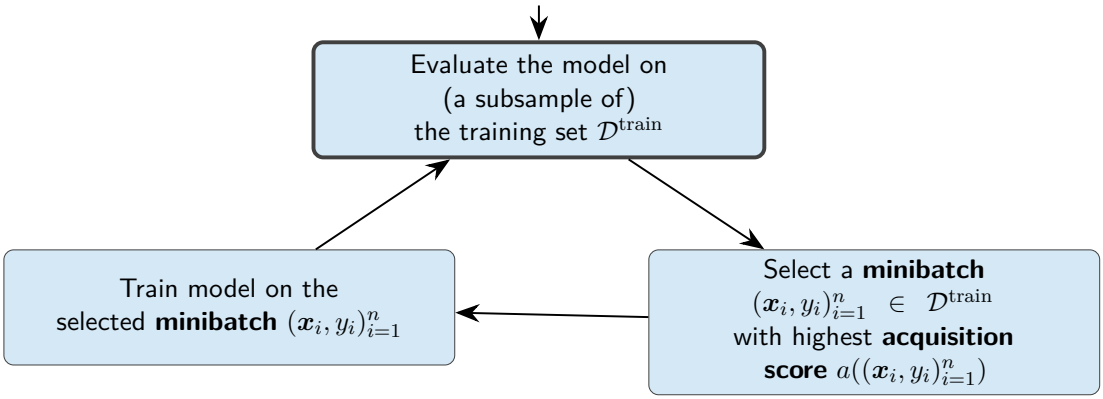
$$a_{\text{Std-Dev}}(\boldsymbol{x}; \mathcal{M}) \triangleq \sum_{y} \sqrt{\mathrm{Var}_{\mathrm{p}(\boldsymbol{\omega})}[\mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})]}. \tag{1.38}$$

This scoring function is also non-negative, and zero variance for the predictions implies a zero expected information gain and thus an uninformative sample.

Information-Theoretic Acquisition Functions. Strategies for constructing such an acquisition function can be based on principled information-theoretic considerations that allow formalizing the notion of the information that will be gained for labelling any given point. These approaches usually require a probabilistic model $\mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})$ for label y given input $\boldsymbol{x}$, where $\boldsymbol{\omega}$ represents a realization of stochastic model parameters $\boldsymbol{\Omega}$; a particularly common choice of model is a Bayesian neural network, wherein $\boldsymbol{\Omega}$ represents the weights and biases.



(a) Active Learning Loop



(b) Active Sampling Loop

Figure 1.3: *(Batch) Active Learning and Active Sampling Loops.* Both active learning and active sampling loops share the same basic structure but differ in the way they are used to train the model. Active learning is used to train a model on a small dataset, while active sampling is used to train a model on a larger dataset. Active learning acquires labels using an expert and adds them to the dataset, while active sampling has access to the labels. Both use an acquisition function to score to select the most informative samples individually or in batches, which is more common in deep learning applications.

1.2.4.1 Bayesian Active Learning

The Bayesian active learning setup consists of an unlabeled dataset $\mathcal{D}^{\text{pool}}$, the current training set $\mathcal{D}^{\text{train}}$, a Bayesian model $\mathcal{M}$ with model parameters $\boldsymbol{\omega} \sim \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$, and output predictions $\mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega}, \mathcal{D}^{\text{train}})$ for data point $\boldsymbol{x}$ and prediction $y \in [C]$ in the classification case. The conditioning of $\boldsymbol{\omega}$ on $\mathcal{D}^{\text{train}}$ expresses that the model has been trained with $\mathcal{D}^{\text{train}}$. Furthermore, an oracle can provide us with the correct label $\tilde{y}$ for a data point in the unlabeled pool $\boldsymbol{x} \in \mathcal{D}^{\text{pool}}$. The goal is to obtain a certain level of prediction accuracy with the least amount of oracle queries. At each acquisition step, a batch of data points $\boldsymbol{x}_{1..K}^{\text{acq},*} = \{\boldsymbol{x}_1^{\text{acq},*}, \ldots, \boldsymbol{x}_K^{\text{acq},*}\}$ is selected using an acquisition function a which scores a candidate batch of unlabeled data points $\boldsymbol{x}_{1..K}^{\text{acq}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\} \subseteq \mathcal{D}^{\text{pool}}$ using the current model parameters $\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$:

$$\boldsymbol{x}_{1..K}^{\text{acq},*} = \underset{\boldsymbol{x}_{1..K}^{\text{acq}} \subseteq \mathcal{D}^{\text{pool}}}{\arg\max}\ a\left(\boldsymbol{x}_{1..K}^{\text{acq}}, \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})\right). \tag{1.39}$$

There are a number of intuitive choices for the acquisition function. We focus on the model uncertainty, which is also known as the expected information gain or BALD [Houlsby et al., 2011], and has proven itself in the context of deep learning [Gal et al., 2017; Shen et al., 2018; Janz et al., 2017]. BALD scores points based on how well their label would inform us about the true model parameter distribution.

1.2.4.2 BALD & Expected Information Gain

BALD (*Bayesian Active Learning by Disagreement*) [Houlsby et al., 2011] uses an acquisition function that estimates the mutual information between the model predictions and the model parameters. It is also referred to as the *Expected Information Gain* [Lindley, 1956] or the *Total Information Gain* [MacKay, 1992c].

The Expected Information Gain (EIG) for $\boldsymbol{\Omega}$ under $\mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})$ was originally introduced in *Bayesian-optimal experiment design (BOED)* [Lindley, 1956; Chaloner and Verdinelli, 1995b; Rainforth et al., 2023] to quantify the utility of data and has a long history [Fedorov, 1972]. The framework of Bayesian experimental design has many applications outside active learning, and in these applications the model parameters are commonly the quantity of interest—Bayesian optimization [Hennig and Schuler, 2012; Hernández-Lobato et al., 2014; Villemonteix et al., 2009] being a notable exception. The EIG in the parameters is thus often a natural acquisition function in BOED:

$$\mathrm{EIG}(\boldsymbol{x}) \triangleq \mathbb{E}_{\mathrm{p}(y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}})}[\mathrm{H}(\mathrm{p}(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})) - \mathrm{H}(\mathrm{p}(\boldsymbol{\Omega} \mid y, \boldsymbol{x}, \mathcal{D}^{\text{train}}))], \tag{1.40}$$

which is equal to the mutual information $\mathrm{I}[\boldsymbol{\Omega}; Y \mid \boldsymbol{x}]$ between the parameters and the label given the data point $\boldsymbol{x}$ [Cavagnaro et al., 2010]. The first term in the expectation is the entropy of the prior distribution over $\boldsymbol{\Omega}$ and the second term is the entropy of the posterior distribution over $\boldsymbol{\Omega}$ given the label of $\boldsymbol{x}$. Intuitively, it measures the expected reduction in uncertainty about $\boldsymbol{\Omega}$ after observing the label of $\boldsymbol{x}$, where $\mathrm{p}(y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}) = \mathbb{E}_{\mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega})}[\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}}]$ is the marginal predictive distribution of the model.

Computing the EIG using above expansion can be difficult as we need to compute the conditional entropy $\mathrm{H}[\boldsymbol{\Omega} \mid Y, \boldsymbol{x}]$—yet this is what many recent approaches effectively attempt to do via the Fisher information. (We do not need to compute the entropy of the parameters $\mathrm{H}[\boldsymbol{\Omega}]$, as it does not depend on the data $\boldsymbol{x}$ and can be ignored.) We will examine this further in §9.

While the EIG focuses on the formulation of the mutual information term as a reduction in model posterior uncertainty given a potential sample, the EIG is also equal to the conditional mutual information between the parameters and the label, $\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$. This equivalent formulation, which focuses on the predictive disagreement, was popularized as BALD in deep active learning by [Gal et al., 2017] as *BALD (Bayesian Active Learning by Disagreement)* [Houlsby et al., 2011] when used with Bayesian neural networks [Neal, 1995]. BALD can be much easier to evaluate by sampling from $\boldsymbol{\Omega}$ without the need for additional Bayesian inference. Originally introduced outside the context of deep learning, the only requirement on the model is that it is Bayesian, but notably BALD is often used even when inference is not precisely Bayesian, for example when using Monte Carlo dropout in a neural network [Gal and Ghahramani, 2016a]. Concretely, BALD is defined following (1.34) as

$$\mathrm{BALD}(\boldsymbol{x}) \triangleq \mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] = \mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}], \tag{1.41}$$

which we already introduced as a means of measuring epistemic uncertainty.

Overall, both formulations express how strongly the model predictions for a given data point and the model parameters are tied, implying that finding out about the true label of data points with high mutual information will also inform us about the true model parameters. Another way to see how BALD captures predictive disagreement is to see that above definition is equivalent to the expected KL divergence between the marginal distribution and the individual distribution, which measures the average disagreement of the posterior predictions:

$$\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] = \mathrm{D}_{\mathrm{KL}}(\mathrm{p}(Y \mid \boldsymbol{x}, \boldsymbol{\Omega}) \parallel \mathrm{p}(Y \mid x)) \tag{1.42}$$

$$= \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})}[\mathrm{D}_{\mathrm{KL}}(\mathrm{p}(Y \mid \boldsymbol{x}, \boldsymbol{\omega}) \parallel \mathrm{p}(Y \mid \boldsymbol{x}))]. \tag{1.43}$$

1.2.5 Batch Active Learning

As two chapters (§4 and §5) of the thesis are concerned with batch active learning, we examine some relevant work here.

Researchers in active learning [Atlas et al., 1989; Settles, 2010] have identified the importance of *batch* acquisition as well as the failures of top-K acquisition using straightforward extensions of single-sample methods in a range of settings including support vector machines [Campbell et al., 2000; Schohn and Cohn, 2000; Brinker, 2003; Guo and Schuurmans, 2007], GMMs [Azimi et al., 2012], and neural networks [Sener and Savarese, 2018; Ash et al., 2020; Baykal et al., 2021]. Many of these methods aim to introduce structured diversity to batch acquisition that accounts for the *interaction* of the points acquired in the learning process.

Maintaining diversity when acquiring a batch of data has been attempted using constrained optimization [Guo and Schuurmans, 2007] and in Gaussian Mixture Models [Azimi et al., 2012]. In active learning of molecular data, the lack of diversity in batches of data points acquired using the BALD objective has been noted by Janz et al. [2017], who propose to resolve it by limiting the number of MC dropout samples and relying on noisy estimates.

In deep learning, another practical aspect that favors batch acquisition over individual point acquisition is that retraining a model can take a substantial amount of time.

One way to extend the formalism of acquisition functions to the batch acquisition case is by scoring batches of samples instead of individual samples: $a\left(\boldsymbol{x}_{1..n}\right)$. In batch active learning, we then want to maximize over possible subsets (of size $\mathtt{K}$):

$$\boldsymbol{x}_{1..\mathtt{K}}^{\text{acq},*} = \underset{\boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}} \subseteq \mathcal{D}^{\text{pool}}}{\arg\max}\ a\left(\boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, \mathcal{M}\right). \tag{1.44}$$

Top-K BALD. BALD was originally intended for acquiring individual data points and immediately retraining the model. Applications of BALD [Gal and Ghahramani, 2016a; Janz et al., 2017] usually acquire the top $\mathtt{K}$ samples. This can be expressed as summing over individual scores:

$$a_{\texttt{BALD}}\left(\boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})\right) = \sum_{i=1}^{\mathtt{K}} \mathrm{BALD}(\boldsymbol{x}_i^{\text{acq}}) \tag{1.45}$$

$$= \sum_{i=1}^{\mathtt{K}} \mathrm{I}[\boldsymbol{\Omega}; Y_i^{\text{acq}} \mid \boldsymbol{x}_i^{\text{acq}}, \mathcal{D}^{\text{train}}], \tag{1.46}$$

and finding the optimal batch for this acquisition function using a greedy algorithm, which reduces to picking the top-K highest-scoring data points.

1.2.6 Active Sampling

Another key problem in deep learning is training efficiency. While large models can exhibit great performance, training them can be time-consuming and expensive. Active sampling is concerned with improving model performance as quickly as possible by selecting the samples to train on from within a larger training set of labeled samples. Unlike in active learning, labels are available, and the goal of active sampling is to select the most informative samples at each step to train on.

Active sampling selects labeled samples to train on *during* training. Like in active learning, candidate batches of labeled samples can be scored via an acquisition function. This results in an active sampling loop which is similar to the active learning loop. However, whereas in active learning the model is usually reset and retrained between iterations [Ash and Adams, 2020], the model weights are, of course, not reset after each iteration. The active sampling loop is depicted in Figure 1.3(b).

Given the conceptual similarities between active learning and active sampling, we will consider how to unify objectives in §9. Recent work has shown that active learning methods can even outperform many active sampling methods without label information in certain circumstances [Park et al., 2022].

Origins. It is not clear where the name active sampling originates from. The author is aware of it being used in meetings with collaborators from Google. A possible reference is Abernethy et al. [2022], which selects samples to train on based on fairness metrics.

Data Subset Selection Methods. Active sampling also encompasses data pruning [Paul et al., 2021; Siddiqui et al., 2023] and core-set selection methods [Campbell and Broderick, 2019, 2018; Mirzasoleiman et al., 2020; Borsos et al., 2020, 2021; Zhou et al., 2023] as well as other data subset selection approaches [Aljundi et al., 2019; Killamsetty et al., 2020, 2021a; Kaushal et al., 2021]: those could be seen as 'single-step' active sampling methods. Guo et al. [2022] contain a good overview of current data subset selection methods.

Information Gain. A natural information-theoretic choice is the *information gain* $\mathrm{I}[\boldsymbol{\Omega}; y \mid \boldsymbol{x}]$ [Sun et al., 2022], which unlike the expected information gain, takes into account the available label. We discuss this notation in §2 in more detail and examine the information gain in §9.

1.3 Thesis Outline

In this thesis, we examine objectives and extensions that follow from information-theoretic intuitions. Similar objectives can be adapted with success for both active learning and active sampling. The thesis is structured as follows:

§2 A Practical & Unified Notation for Information Quantities. We define information quantities using a notation that allows us to take into account outcomes as well. Usually, information quantities are only explicitly considered between unobserved random variables. The exception to that is the point-wise mutual information, which is well-known in NLP. Our notation unifies the mutual information and the point-wise mutual information (and the information gain and information surprise). This chapter is entirely based on Kirsch and Gal [2021]:

Andreas Kirsch and Yarin Gal. A Practical & Unified Notation for Information-Theoretic Quantities in ML. *arXiv preprint*, 2021.

§3 Single Forward-Pass Aleatoric and Epistemic Uncertainty. Epistemic uncertainty is important as an informativeness score in active learning. We examine aleatoric and epistemic uncertainty in more detail, and based on several simple but crucial high-level observations, we propose a new baseline for uncertainty quantification using single forward-pass deep neural networks, e.g. deterministic neural networks instead of deep ensembles or similar. We draw a connection between feature-space density and informativeness (epistemic uncertainty) and show that with proper inductive biases, our simple approach can quantify epistemic uncertainty well with competitive results in active learning (and out-of-distribution detection) without having to be Bayesian. This chapter redrafts Mukhoti et al. [2023]:

Jishnu Mukhoti*, Andreas Kirsch*, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023,

with additional figures, explanations, and results (§3 and §3.1.3, Figure 3.1, and Proposition 3.2).

§4 Diverse Batch Acquisition for Bayesian Active Learning. Active learning is often extended to batch active learning by greedily selecting the top-`K` samples that individually have the highest informativeness score. We note that this is not a principled approach and can lead to worse active learning performance as it does not take dependencies and redundancies between the samples into account. We derive BatchBALD, a principled extension of the BALD acquisition score for batch acquisition, which avoids this issue. The employed techniques and insights are not limited to BALD and are used throughout the thesis. This chapter extends Kirsch et al. [2019]:

Andreas Kirsch*, Joost van Amersfoort*, and Yarin Gal. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *Advances in Neural Information Processing Systems*, 2019,

with additional figures (Figures 4.3 and 4.5).

§5 Stochastic Batch Acquisition for Deep Active Learning. BatchBALD has difficulties in scaling to higher batch acquisition sizes, both due to the computational cost and the scores becoming too uniform at higher batch acquisition sizes. Instead, we take a different approach, and we examine how progressive acquisitions in active learning change the EIG scores and their effect on top-`K` batch acquisitions. Based on this, we examine a simple stochastic baseline that avoids the pathologies of top-`K` acquisition by adding Gumbel noise to the individual scores. This very simple baseline is surprisingly strong and also much cheaper to compute than the principled extension of the EIG to batch acquisition in §4. This chapter is entirely based on Kirsch et al. [2023]:

Andreas Kirsch*, Sebastian Farquhar*, Parmida Atighehchian, Andrew Jesson, Frederic Branchaud-Charron, and Yarin Gal. Stochastic Batch Acquisition for Deep Active Learning. *Transactions on Machine Learning Research*, 2023.

§6 Marginal and Joint Cross-Entropies & Predictives. Focusing on joint predictives rather than marginal predictives can highlight the potential of Bayesian deep learning in real-world applications. In this chapter, we discuss online Bayesian inference, which allows making predictions while taking into account additional data without retraining, and propose new challenging evaluation settings using active learning and active sampling. By examining marginal and joint predictives, their respective cross-entropies, and their role in offline and online learning, we highlight previously unidentified gaps in current research and emphasize the need for better approximate joint predictives. This chapter builds on insights from Wen et al. [2021] and Osband et al. [2022b], and we suggest further experiments to explore the feasibility of current BDL inference techniques in high-dimensional parameter spaces. This chapter is entirely based on Kirsch et al. [2022]:

Andreas Kirsch, Jannik Kossen, and Yarin Gal. Marginal and Joint Cross-Entropies & Predictives for Online Bayesian Inference, Active Learning, and Active Sampling. *arXiv preprint*, 2022.

§7 Prediction- & Distribution-Aware Bayesian Active Learning. The EIG does not take the target distribution of inputs into account at all. In other words (and if the target distribution matches the pool set distribution), it does not make use of the unlabeled data to guide the sample selection. To account for this, we derive the *expected predictive information gain (EPIG)*, an information quantity, which takes the data distribution into account. We show that it equivalently measures the expected reduction in generalization loss on the target distribution. We also examine and compare to the *joint expected predictive information gain (JEPIG)*, a related quantity, which we connect to Bayesian model selection. This chapter extends Kirsch et al. [2021] and Smith et al. [2023]:

Andreas Kirsch, Tom Rainforth, and Yarin Gal. Active Learning under Pool Set Distribution Shift and Noisy Data. *arXiv preprint*, 2021,

Freddie Bickford Smith*, Andreas Kirsch*, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. Prediction-Oriented Bayesian Active Learning. *International Conference on Artificial Intelligence and Statistics*, 2023,

with additional and reworked sections, examples, and figures (Example 7.1, §7.2.2, §7.4.1 and §7.4.2, Figure 7.3, and Proposition 7.2).

§8 Prioritized Data Selection during Training. For active sampling (label-aware active learning), we examine an adaptation of the previous two acquisition functions that take the labels into account: the *(joint) predictive information gain—(J)PIG*. Training on web-scale data can take months. But most computation and time is wasted on redundant and noisy points that are already learned or not learnable.

To accelerate training, based on JPIG, we perform a series of approximations and introduce the *Reducible Holdout Loss Selection* (RHO-LOSS) in a non-Bayesian setting using two non-Bayesian (deterministic) models for active sampling. RHO-LOSS is a simple but principled technique which selects approximately those points for training that most reduce the model's generalization loss and trains in far fewer steps than prior art, improves accuracy, and speeds up training on a wide range of datasets, hyperparameters, and architectures (MLPs, CNNs, and BERT). On a large web-scraped image dataset (Clothing-1M), RHO-LOSS trains in 18x fewer steps and reaches 2% higher final accuracy than uniform data shuffling. This chapter is based on Mindermann et al. [2022]:

> Sören Mindermann*, Jan Markus Brauner*, Muhammed Razzak*, Mrinank Sharma*, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized Training on Points that are Learnable, Worth Learning, and not yet Learnt. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022,

with a reworked theory section (§8.2).

§9 Unifying Approaches in Active Learning and Active Sampling. Many approaches in data subset selection use Fisher information, Hessians, similarity matrices based on gradients, or the gradient length to estimate how informative data is for a model's training. Are these different approaches connected, and if so, how? We revisit the fundamentals of Bayesian optimal experiment design and show that these recently proposed methods can be understood as approximations to the information-theoretic quantities examined in this thesis: EIG/BALD and (J)EPIG for active learning, and the information gain (IG) and (J)PIG for active sampling. We develop a comprehensive set of approximations using Fisher information and the observed information and derive a unified framework that connects above seemingly disparate literature. Although Bayesian methods are often seen as separate from non-Bayesian ones, the sometimes fuzzy notion of "informativeness" expressed in various non-Bayesian objectives leads to the same couple of information quantities, which were, in principle, already known by Lindley [1956] and MacKay [1992b]. This chapter is entirely based on Kirsch and Gal [2022b]:

> Andreas Kirsch and Yarin Gal. Unifying Approaches in Active Learning and Active Sampling via Fisher Information and Information-Theoretic Quantities. *Transactions on Machine Learning Research*, 2022b.

§10 Black-Box Batch Active Learning for Regression. §9 show that different active learning methods approximate the same information-theoretic quantities using two different perspectives:

- Bayesian methods (including deep ensembles) often use (sampled) predictions over the parameter distribution to approximate information quantities, while
- non-Bayesian methods often use the weight space (using the score, i.e. log loss gradients) to approximate the same information quantities.

The difference between these two perspectives can be viewed as being between *white-box approaches*, which are limited to differentiable parametric models (weight

space) and *black-box approaches*, which only use model predictions (prediction space). White-box methods score unlabeled points using acquisition functions based on model embeddings or first- and second-order derivatives. We utilize recent kernel-based approaches and turn a wide range of existing state-of-the-art white-box batch active learning methods (BADGE, BAIT, LCMD) into black-box approaches. We demonstrate the effectiveness of our approach through extensive experimental evaluations on regression datasets, achieving surprisingly strong performance compared to white-box approaches for deep learning models. This chapter is entirely based on Kirsch [2023a]:

> Andreas Kirsch. Black-Box Batch Active Learning for Regression. *Transactions on Machine Learning Research*, 2023a.

Appendices In the appendix, we provide additional details on the information-theoretic quantities and the approximations used in this thesis. We also include additional works that are related to the topics in this thesis and reproducibility analyses of several existing works.

Appendix A Causal-BALD: Deep Bayesian Active Learning of Outcomes to Infer Treatment-Effects from Observational Data. As a different application of combining information-theoretic intuitions and active learning, we develop acquisition functions for estimating the conditional average treatment effects from observational data for causal active learning. Unlike the previous applications, it does not apply to classification tasks but regression tasks. This chapter is entirely based on Jesson et al. [2021]:

> Andrew Jesson*, Panagiotis Tigas*, Joost van Amersfoort, Andreas Kirsch, Uri Shalit, and Yarin Gal. Causal-BALD: Deep Bayesian Active Learning of Outcomes to Infer Treatment-Effects from Observational Data. In *Advances in Neural Information Processing Systems*, 2021.

Appendix B Reproducibility Analysis. We also include reproducibility analyses of several papers that provide additional relevant insights (which are connected to themes in this thesis):

Appendix B.1 Deep Learning on a Data Diet. We reproduce parts of Paul et al. [2021]. This chapter is entirely based on Kirsch [2023b]:

> Andreas Kirsch. Does "Deep Learning on a Data Diet" reproduce? Overall yes, but GraNd at Initialization does not. *Transactions on Machine Learning Research*, 2023b.

Appendix B.2 A Note on "Assessing Generalization of SGD via Disagreement" We examine details of Jiang et al. [2022]. This chapter is entirely based on Kirsch and Gal [2022a]:

> Andreas Kirsch and Yarin Gal. A Note on "Assessing Generalization of SGD via Disagreement". *Transactions on Machine Learning Research*, 2022a.

Appendix B.3 Dirichlet Model of a Deep Ensemble's Softmax Predictions We examine how well Dirichlet distributions can model the predictions of deep ensembles and their members to approximate information quantities

we care about and to sample from the posterior predictive distribution. This chapter is entirely based on Mukhoti et al. [2023]:

> Jishnu Mukhoti*, Andreas Kirsch*, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023,

and extends it with additional results.

The limits of my language mean the limits of my world.

Ludwig Wittgenstein

2

A Practical & Unified Notation for Information Quantities

In §1.2.1, we have introduced well-known information-theoretic quantities using a more consistent notation. Now, we further canonically extend the definitions to tie random variables to specific observed outcomes, e.g. $X = x$. We will use this extensively in the following chapters. We refer to X when we have $X = x$ in an expression as *tied random variable* as it is *tied* to an outcome. If we mix *(untied) random variables* and *tied random variables*, we define $\mathrm{H}[\,\cdot\,]$ as an operator which takes an expectation of Shannon's information content for the given expression over the (untied) random variables conditioned on the tied outcomes. For example, $\mathrm{H}[X, Y = y \mid Z, W = w] = \mathbb{E}_{\mathrm{p}(X,Z \mid y,w)} \mathrm{H}(\mathrm{p}(x, y \mid z, w))$ following this notation. We generally shorten $Y = y$ to y when the connection is clear from context—except for the datasets $\mathcal{D}^{\text{pool}}, \mathcal{D}^{\text{train}}$, etc., which are sets of outcomes (either only containing inputs $\boldsymbol{x}$ when unlabeled or containing both inputs $\boldsymbol{x}$ and targets y). Similarly, we have $\mathrm{H}(\mathrm{p}(X \mid y) \parallel \mathrm{q}(X \mid y)) = \mathbb{E}_{\mathrm{p}(x \mid y)} \mathrm{H}(\mathrm{q}(x \mid y))$.

As a memory hook for the reader, lower-case letters are always used for tied random variables and upper-case letters for (untied) random variables over which we take an expectation. This makes it easy to differentiate between the two cases and write down the actual expressions.

Importantly, the definitions above maintain the identities $\mathrm{H}[X, Y] = \mathbb{E}_{\mathrm{p}(x)} \mathrm{H}[x, Y] = \mathbb{E}_{\mathrm{p}(y)} \mathrm{H}[X, y]$, which is the motivation behind these extensions. Figure 2.1 provides an overview over the quantities for two random variables X and Y when $Y = y$ is observed. We define everything in detail below and provide intuitions.

Definition 2.1. Given random variables X and Y and outcome y, we define:

$$\mathrm{H}[y] \triangleq \mathrm{H}(\mathrm{p}(y)) \tag{2.1}$$

$$\mathrm{H}[X, y] \triangleq \mathbb{E}_{\mathrm{p}(x \mid y)} \mathrm{H}[x, y] = \mathbb{E}_{\mathrm{p}(x \mid y)} \mathrm{H}(\mathrm{p}(x, y)) \tag{2.2}$$

$$\mathrm{H}[X \mid y] \triangleq \mathbb{E}_{\mathrm{p}(x \mid y)} \mathrm{H}[x \mid y] = \mathbb{E}_{\mathrm{p}(x \mid y)} \mathrm{H}(\mathrm{p}(x \mid y)) \tag{2.3}$$

$$\mathrm{H}[y \mid X] \triangleq \mathbb{E}_{\mathrm{p}(x \mid y)} \mathrm{H}[y \mid x] = \mathbb{E}_{\mathrm{p}(x \mid y)} \mathrm{H}(\mathrm{p}(y \mid x)), \tag{2.4}$$

where we have shortened $Y = y$ to y.

Again, $\mathrm{H}[y], \mathrm{H}[X, y]$ are shorthands for $\mathrm{H}(\mathrm{p}(y))$, $\mathrm{H}(\mathrm{p}(X, y))$, and so on.

The intuition from information theory behind these definitions is that, e.g., $\mathrm{H}[X, y]$ measures the average length of transmitting X and Y together when $Y = y$ unbeknownst

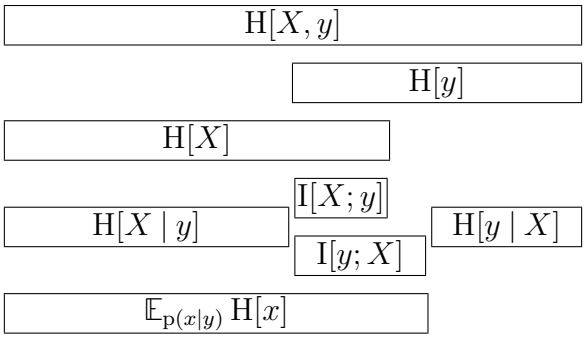


Figure 2.1: *The relationship between joint entropy* $\mathrm{H}[X, y]$*, entropies* $\mathrm{H}[X], \mathrm{H}[y]$*, conditional entropies* $\mathrm{H}[X \mid y], \mathrm{H}[y \mid X]$*, information gain* $\mathrm{I}[X; y]$ *and surprise* $\mathrm{I}[y; X]$ *when* $Y = y$ *is observed.* We include $\mathbb{E}_{\mathrm{p}(x|y)} \mathrm{H}[x]$ to visualize Proposition 2.1. The figure follows Figure 8.1 in MacKay [2003].

to the sender and receiver, and $\mathrm{H}[y \mid X]$ measures how much additional information needs to be transferred on average for the receiver to learn y when it already knows $X \,|\, y$.

From above definition, we also have $\mathrm{H}[x, y] = \mathrm{H}(\mathrm{p}(x, y))$ and $\mathrm{H}[x \mid y] = \mathrm{H}(\mathrm{p}(x \mid y))$. Beware, however, that while we have $\mathrm{H}[X \mid y] = \mathrm{H}[X, y] - \mathrm{H}[y]$, for $\mathrm{H}[y \mid X]$, there is no such equality for $\mathrm{H}[y \mid X]$:

Proposition 2.1. *Given random variables* X *and* Y *and outcome* y*, we generally have:*

$$\mathrm{H}[X \mid y] = \mathrm{H}[X, y] - \mathrm{H}[y] \tag{2.5}$$

$$\begin{aligned} \mathrm{H}[y \mid X] &= \mathrm{H}[X, y] - \mathbb{E}_{\mathrm{p}(x|y)} \mathrm{H}[x] \\ &\neq \mathrm{H}[X, y] - \mathrm{H}[X], \end{aligned} \tag{2.6}$$

Proof. $\mathrm{H}[X \mid y] = \mathrm{H}[X, y] - \mathrm{H}[y]$ follows immediately from the definitions. $\mathrm{H}[y \mid X] \neq \mathrm{H}[X, y] - \mathrm{H}[X]$ follows because, generally, $\mathbb{E}_{\mathrm{p}(x|y)} \mathrm{H}[x] \neq \mathrm{H}[X]$ when $\mathrm{p}(x|y) \neq \mathrm{p}(x)$. E.g., for X and Y only taking binary values, 0 or 1, let[1] $\mathrm{p}(x, y) = \frac{1}{3}\mathbb{1}_{\{x=0=y\}}$, then $\mathbb{E}_{\mathrm{p}(x|y)} \mathrm{H}[x] = \log\left(\frac{3}{2}\right) \neq \log\left(\frac{3\sqrt[3]{2}}{2}\right) = \mathrm{H}[X]$. □

There are two common, sensible quantities we can define when we want to consider the information overlap between a random variable and an outcome: the *information gain*, also known as *specific information* and the *surprise* [DeWeese and Meister, 1999; Butts, 2003]. These two quantities are usually defined separately in the cognitive sciences and neuroscience [Williams, 2011]; however, we can unify them after relaxing the symmetry of the mutual information as done above:

Definition 2.2. Given random variables X and Y and outcome y for Y, we define the *information gain* $\mathrm{I}[X; y]$ and the *surprise* $\mathrm{I}[y; X]$ as:

$$\mathrm{I}[X; y] \triangleq \mathrm{H}[X] - \mathrm{H}[X \mid y] \tag{2.7}$$

$$\mathrm{I}[y; X] \triangleq \mathrm{H}[y] - \mathrm{H}[y \mid X]. \tag{2.8}$$

[1] See also https://colab.research.google.com/drive/1HvLXUMQYcxMGZ4S_a00xddGmfz0IHaR3.

This unifying definition is novel to the best of our knowledge. It works by breaking the symmetry that otherwise exists for the regular and point-wise mutual information.

Note that the surprise can also be expressed as $\mathrm{I}[y;X] = \mathrm{D}_{\mathrm{KL}}(\mathrm{p}(X \mid y) \parallel \mathrm{p}(X))$. For example, this is done in Bellemare et al. [2016]—even though the paper mistakenly calls this surprise an information gain when it is not (in our sense).

We enumerate a few equivalent ways of writing the mutual information and surprise—the information gain has no such equivalences. This can be helpful to spot these quantities in the wild.

Proposition 2.2. *We have*

$$\mathrm{I}[X;Y] = \mathrm{D}_{\mathrm{KL}}(\mathrm{p}(X,Y) \parallel \mathrm{p}(X)\,\mathrm{p}(Y)) \tag{2.9}$$
$$\mathrm{I}[y;X] = \mathbb{E}_{\mathrm{p}(x|y)}\,\mathrm{I}[y;x] \tag{2.10}$$
$$= \mathbb{E}_{\mathrm{p}(x|y)}[\mathrm{H}[x]] - \mathrm{H}[X \mid y] \tag{2.11}$$
$$= \mathrm{D}_{\mathrm{KL}}(\mathrm{p}(X \mid y) \parallel \mathrm{p}(X)). \tag{2.12}$$

The information gain $\mathrm{I}[X;y]$ for X given y measures the reduction in uncertainty about $\mathrm{H}[X]$ when we observe y. $\mathrm{H}[X]$ is the uncertainty about the true X that we want to learn: the entropy quantifies the amount of additional information that we need to transmit to fix X, and similarly $\mathrm{H}[X \mid y]$ quantities the additional information we need to transmit to fix X once y is known both to the sender and the receiver [Lindley, 1956]. On the other hand, the surprise $\mathrm{I}[y;X]$ of y for X measures how much the posterior $X \mid y$ lies in areas where $\mathrm{p}(x)$ was small before observing y [DeWeese and Meister, 1999].

An important difference between the two is that the information gain can be *chained* while the surprise cannot:

Proposition 2.3. *Given random variables X, Y_1, and Y_2 and outcomes y_1 and y_2 for Y_1 and Y_2, respectively, we have:*

$$\mathrm{I}[X;y_1,y_2] = \mathrm{I}[X;y_1] + \mathrm{I}[X;y_2 \mid y_1] \tag{2.13}$$
$$\mathrm{I}[y_1,y_2;X] \neq \mathrm{I}[y_1;X] + \mathrm{I}[y_2;X \mid y_1]. \tag{2.14}$$

Proof. We have

$$\begin{aligned}\mathrm{I}[X;y_1,y_2] &= \mathrm{H}[X] - \mathrm{H}[X \mid y_1,y_2] \\ &= \mathrm{H}[X] - \mathrm{H}[X \mid y_1] + \mathrm{H}[X \mid y_1] - \mathrm{H}[X \mid y_1,y_2] \\ &= \mathrm{I}[X;y_1] + \mathrm{I}[X;y_2 \mid y_1],\end{aligned}$$

while

$$\begin{aligned}\mathrm{I}[y_1,y_2;X] &= \mathbb{E}_{\mathrm{p}(x|y_1,y_2)}\,\mathrm{I}[y_1,y_2;x] \\ &= \underbrace{\mathbb{E}_{\mathrm{p}(x|y_1,y_2)}\,\mathrm{I}[y_1;x]}_{\neq\ \mathbb{E}_{\mathrm{p}(x|y_1)}\,\mathrm{I}[y_1;x] = \mathrm{I}[y_1;X]} \\ &\quad + \underbrace{\mathbb{E}_{\mathrm{p}(x|y_1,y_2)}\,\mathrm{I}[y_2;x \mid y_1]}_{=\mathrm{I}[y_2;X \mid y_1]}.\end{aligned}$$

That is, generally, $\mathbb{E}_{\mathrm{p}(x|y_1,y_2)}\,\mathrm{I}[y_1;x] \neq \mathrm{I}[y_1;X]$. To conclude the proof, we instantiate $\mathrm{p}(x|y_1,y_2) \neq \mathrm{p}(x|y_1)$: for X, Y_1, and Y_2 taking binary values 0, 1 only, let $\mathrm{p}(y_1) = \frac{1}{2}$, $\mathrm{p}(x,y_2|$

$y_1 = 0) = \frac{1}{4}$, $\mathrm{p}(x \mid y_2 = 0, y_1 = 1) = \frac{1}{2}$, $\mathrm{p}(x = 0 \mid y_2 = 1, y_1 = 1) = 1$. Then $\mathbb{E}_{\mathrm{p}(x|y_1,y_2)} \mathrm{I}[y_1; x] = \log\left(\frac{2\sqrt{3}\sqrt[4]{5}}{5}\right) \neq \log\left(\frac{6}{5}\right) = \mathrm{I}[y_1; X]$ for $y_1 = 1, y_2 = 1$ as the reader can easily verify[2]. □

However, both quantities do chain in their (untied) random variables:

Proposition 2.4. *Given random variables X_1, X_2, Y, and outcome y for Y:*

$$\mathrm{I}[X_1, X_2; y] = \mathrm{I}[X_1; y] + \mathrm{I}[X_2; y \mid X_1] \tag{2.15}$$
$$\mathrm{I}[y; X_1, X_2] = \mathrm{I}[y; X_1] + \mathrm{I}[y; X_2 \mid X_1]. \tag{2.16}$$

Proof. We have

$$\begin{aligned}
&\mathrm{I}[X_1; y] + \mathrm{I}[X_2; y \mid X_1] = \\
&\quad = \mathrm{H}[X_1] - \mathrm{H}[X_1 \mid y] + \mathrm{H}[X_2 \mid X_1] + \mathrm{H}[X_2 \mid X_1, y] \\
&\quad = \underbrace{\mathrm{H}[X_1] + \mathrm{H}[X_2 \mid X_1]}_{=\mathrm{H}[X_1, X_2]} - \underbrace{(\mathrm{H}[X_1 \mid y] + \mathrm{H}[X_2 \mid X_1, y])}_{=\mathrm{H}[X_1, X_2 \mid y]} \\
&\quad = \mathrm{I}[X_1, X_2; y].
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
&\mathrm{I}[y; X_1] + \mathrm{I}[y; X_2 \mid X_1] = \\
&\quad = \mathrm{H}[y] - \mathrm{H}[y \mid X_1] + \mathrm{H}[y \mid X_1] - \mathrm{H}[y \mid X_1, X_2] \\
&\quad = \mathrm{H}[y] - \mathrm{H}[y \mid X_1, X_2] \\
&\quad = \mathrm{I}[y; X_1, X_2].
\end{aligned}$$

□

These extensions of the mutual information are canonical as they permute with taking expectations over tied variables to obtain the regular (untied) quantities:

Proposition 2.5. *For random variables X and Y:*

$$\mathrm{I}[X; Y] = \mathbb{E}_{\mathrm{p}(y)} \mathrm{I}[X; y] = \mathbb{E}_{\mathrm{p}(y)} \mathrm{I}[y; X] = \mathbb{E}_{\mathrm{p}(x,y)} \mathrm{I}[x, y]. \tag{2.17}$$

Proof. Follows immediately from substituting the definitions. □

Likewise, when all random variables are tied to a specific outcome, the quantities behave as expected:

Proposition 2.6. *For random variables X, Y, Y_1 and Y_2:*

$$\mathrm{I}[X; Y] = \mathrm{I}[Y; X], \text{ and} \tag{2.18}$$
$$\mathrm{I}[x; y] = \mathrm{I}[y; x]; \tag{2.19}$$
$$\mathrm{I}[X; Y_1, Y_2] = \mathrm{I}[X; Y_1] + \mathrm{I}[X; Y_1 \mid Y_2], \text{ and} \tag{2.20}$$
$$\mathrm{I}[x; y_1, y_2] = \mathrm{I}[x; y_1] + \mathrm{I}[x; y_2 \mid y_1]. \tag{2.21}$$

[2] See also https://colab.research.google.com/drive/1gn6oQohRMqXKEhyCogiVDcx1VZFkShaQ.

Proof. The only interesting equality is $\mathrm{I}[x; y_1, y_2] = \mathrm{I}[x; y_1] + \mathrm{I}[x; y_2 \mid y_1]$:

$$\begin{aligned}\mathrm{I}[x; y_1] + \mathrm{I}[x; y_2 \mid y_1] &= \mathrm{H}\left(\frac{\mathrm{p}(x)\ \mathrm{p}(y_1)}{\mathrm{p}(x, y_1)} \frac{\mathrm{p}(x, y_1)\ \mathrm{p}(y_1, y_2)\ \mathrm{p}(y_1)}{\mathrm{p}(y_1)\ \mathrm{p}(y_1)\,\mathrm{p}(x, y_1, y_2)}\right) \\ &= \mathrm{H}\left(\frac{\mathrm{p}(x)\ \mathrm{p}(y_1, y_2)}{\mathrm{p}(x, y_1, y_2)}\right) \\ &= \mathrm{I}[x; y_1, y_2].\end{aligned}$$

□

We can extend this to triple mutual information terms by adopting the extension $\mathrm{I}[X; Y; Z] = \mathrm{I}[X; Y] - \mathrm{I}[X; Y \mid Z]$ [Yeung, 2008] for outcomes as well: $\mathrm{I}[X; Y; z] = \mathrm{I}[X; Y] - \mathrm{I}[X; Y \mid z]$, which also works for higher-order terms.

Overall, for the reader, there will be little surprise when working with the fully pointwise information-theoretic quantities, that is, when all random variables are observed. But the mixed ones require more care. We refer the reader back to Figure 2.1 to recall the relationships which also provide intuitions for the inequalities we will examine next.

Inequalities. We review some well-known inequalities first:

Proposition 2.7. *For random variables X and Y, we have:*

$$\mathrm{I}[X; Y] \geq 0 \tag{2.22}$$
$$\mathrm{H}[X] \geq \mathrm{H}[X \mid Y], \tag{2.23}$$

and if X is a discrete random variable, we also have:

$$\mathrm{H}[X] \geq 0 \tag{2.24}$$
$$\mathrm{I}[X; Y] \leq \mathrm{H}[X]. \tag{2.25}$$

Proof. The first two statements follow from:

$$\begin{aligned}\mathrm{H}[X] - \mathrm{H}[X \mid Y] &= \mathrm{I}[X; Y] \\ &= \mathrm{D}_{\mathrm{KL}}(\mathrm{p}(X, Y) \parallel \mathrm{p}(X)\,\mathrm{p}(Y)) \\ &\geq 0. \end{aligned} \tag{2.26}$$

The third statement follows from the monotony of the expectation and $\mathrm{p}(x) \leq 1$ for all x. □

For mixed outcomes we find similar inequalities:

Proposition 2.8. *For random variables X and Y with outcome y, we have:*

$$\mathrm{I}[y; X] \geq 0 \tag{2.27}$$
$$\mathrm{H}[y] \geq \mathrm{H}[y \mid X] \tag{2.28}$$
$$\mathbb{E}_{\mathrm{p}(x|y)\mathrm{H}[x]} \geq \mathrm{H}[X \mid y], \tag{2.29}$$

and if Y is a discrete random variable, we also have:

$$\mathrm{H}[y \mid X], \mathrm{H}[y] \geq 0 \tag{2.30}$$
$$\mathrm{I}[y; X] \leq \mathrm{H}[y], \tag{2.31}$$

and if X is also a discrete random variable, we gain:

$$\mathrm{I}[y; X] \leq \mathbb{E}_{\mathrm{p}(x|y)}\, \mathrm{H}[x]. \tag{2.32}$$

Proof. Again, the first two statements follow from:

$$\begin{aligned} \mathrm{H}[y] - \mathrm{H}[y \mid X] &= \mathrm{I}[y; X] \\ &= \mathbb{E}_{\mathrm{p}(x|y)}\, \mathrm{I}[y; x] \\ &= \mathbb{E}_{\mathrm{p}(x|y)}[\mathrm{H}[x] - \mathrm{H}[x \mid y]] \qquad (2.33) \\ &= \mathrm{D}_{\mathrm{KL}}(\mathrm{p}(X \mid y) \parallel \mathrm{p}(X)) \\ &\geq 0. \qquad (2.34) \end{aligned}$$

The third statement follows from Equation 2.33 above as $0 \leq \mathbb{E}_{\mathrm{p}(x|y)}[\mathrm{H}[x] - \mathrm{H}[x \mid y]] = \mathbb{E}_{\mathrm{p}(x|y)}\, \mathrm{H}[x] - \mathrm{H}[X \mid y]$. The fourth statement follows from $\mathrm{p}(y \,|\, x) \leq 1$ when Y is a discrete random variable, and thus $\mathrm{H}[y \,|\, X] \geq 0$ due to the monotony of the expectation. The fifth statement follows from the fourth statement and $\mathrm{I}[y; X] = \mathrm{H}[y] - \mathrm{H}[y \mid X] \leq \mathrm{H}[y]$. Finally, if X is a discrete random variable as well, we also have $\mathrm{H}[X \mid y] \geq 0$, and thus

$$\mathrm{I}[y; X] = \mathbb{E}_{\mathrm{p}(x|y)}[\mathrm{H}[x] - \mathrm{H}[X \mid y]] \leq \mathbb{E}_{\mathrm{p}(x|y)}\, \mathrm{H}[x].$$

□

Note that there are no such bounds for $\mathrm{I}[X; y]$, $\mathrm{H}[X \mid y]$ and $\mathrm{H}[y \mid X]$.

Corollary 2.9. *We have* $\mathrm{I}[y; X] = 0$ *exactly when* $\mathrm{p}(x \mid y) = \mathrm{p}(x)$ *for all* x *for given* y.

Proof. This follows from $0 = \mathrm{I}[y; X] = \mathrm{D}_{\mathrm{KL}}(\mathrm{p}(X \mid y) \parallel \mathrm{p}(X))$ exactly when $\mathrm{p}(x \mid y) = \mathrm{p}(x)$. □

In particular, there is a misleading intuition that the information gain $\mathrm{I}[X; y] = \mathrm{H}[X] - \mathrm{H}[X \,|\, y]$ ought to be non-negative for any y. This is not true. This intuition may exist because in many cases when we look at posterior distributions, we only model the mean and assume a fixed variance. The uncertainty around the mean does indeed reduce with additional observations; however, the uncertainty around the variance might not. The reader is invited to experiment with a normal distribution with known mean and compute the information gain on the variance depending on new observations.

In a sense, the information-theoretic surprise is much better behaved than the information gain because we can bound it in various ways, which does not seem possible for the information gain. The (expected) information gain is a more useful quantity though for active sampling, active learning and Bayesian optimal experimental design. Thus, unified notation that includes and differentiates between both quantities is beneficial.

2.1 Example Application: Stirling's Approximation for Binomial Coefficients

In MacKay [2003] on page 2, the following simple approximation for a binomial coefficient is introduced:

$$\log \binom{N}{r} \simeq (N - r) \log \frac{N}{N - r} + r \log \frac{N}{r}. \qquad (2.35)$$

We will derive this result using the proposed extension to observed outcomes as it allows for an intuitive deduction. Moreover, we will see that this allows us to use other tools from probability theory to estimate the approximation error.

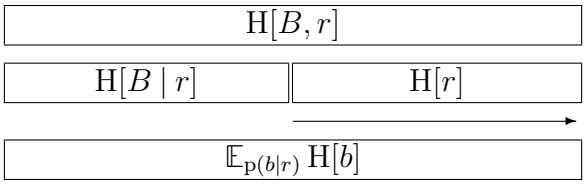


Figure 2.2: *The relationship between the information quantities used in §2.1.* B is the joint of the binomial random variables, R is the number of successes in B with observed outcome r. The arrow below $\mathrm{H}[r]$ symbolizes that we minimize $\mathrm{H}[r]$ by optimizing the success probability ρ to close the gap between $\mathbb{E}_{\mathrm{p}(b|r)}\,\mathrm{H}[b]$ and $\mathrm{H}[B \mid r]$.

Setup. Let $B_1, \ldots, B_N$ be N Bernoulli random variables with success probability p, and let B be the joint of these random variables.

Further, let R be the random variable that counts the number of successes in B. R follows a Binomial distribution with success probability ρ and N trials.

Main Idea. For a given outcome r of R, we have:

$$\mathrm{H}[B, r] = \mathrm{H}[B \mid r] + \mathrm{H}[r] \geq \mathrm{H}[B \mid r], \tag{2.36}$$

as $\mathrm{H}[\cdot]$ is non-negative for discrete random variables. We will examine this inequality to obtain the approximation in Equation 2.35.

Note that $\mathrm{H}[B \mid r]$ is the additional number of bits needed to encode B when the number of successes is already known. Similarly, $\mathrm{H}[B, r]$ is the number of bits needed to encode both B and R under the circumstance that $R = r$.

Determining $\mathrm{H}[B, r]$**.** R is fully determined by B, and thus we have $\mathrm{H}[B, R] = \mathrm{H}[B]$ and hence[3]:

$$\mathrm{H}[B, r] = \mathbb{E}_{\mathrm{p}(b|r)}\,\mathrm{H}[b]. \tag{2.37}$$

$\mathbb{E}_{\mathrm{p}(b|r)}\,\mathrm{H}[b]$ is the expected number of bits needed to transmit the outcome b of B when r is given. When we encode B, we do not know r upfront, so we need to transmit N Bernoulli outcomes. Hence, we need to transmit r successes and $N - r$ failures. Given the success probability ρ, the optimal message length for this is:

$$\mathbb{E}_{\mathrm{p}(b|r)}\,\mathrm{H}[b] = r\ \mathrm{H}(\rho) + (N - r)\ \mathrm{H}(1 - \rho) \tag{2.38}$$

$$= -r \log \rho - (N - r) \log(1 - \rho). \tag{2.39}$$

All this is visualized in Figure 2.2.

Alternative Argument. We can also look at the terms $\mathrm{H}[B \mid r] + \mathrm{H}[r]$ separately. We have

$$\mathrm{H}[r] = -\log \mathrm{p}(r) = -\log \left(\binom{N}{r} \rho^r \, (1 - \rho)^{N-r} \right), \tag{2.40}$$

and

$$\mathrm{H}[B \mid r] = -\,\mathbb{E}_{\mathrm{p}(b|r)} \log \mathrm{p}(b \mid r) = \log \binom{N}{r}. \tag{2.41}$$

[3]This also follows immediately from $\mathrm{H}[R \mid B] = 0 \implies \forall r : \mathrm{H}[r \mid B] = 0$.

The former follows from R being binomial distributed. For the latter, we observe that we need to encode B while knowing r already. Given r, $\mathrm{p}(b \mid r) = \text{const}$ for all valid b. There are $\binom{N}{r}$ possible b for fixed r. Hence, we can simply create a table with all possible configurations with r successes. There are $\binom{N}{r}$ many. We then encode the index into this table.

Each configuration with r successes has an equal probability of happening, so we have a uniform discrete distribution with entropy $\log \binom{N}{r}$ and obtain the same result.

Determining ρ. We already have

$$\begin{aligned} \mathrm{H}[B \mid r] + \mathrm{H}[r] &= -r \log \rho - (N - r) \log(1 - \rho) \\ &\geq \log \binom{N}{r} = \mathrm{H}[B \mid r]. \end{aligned} \tag{2.42}$$

How do we make this inequality as tight as possible?

We need to minimize the gap $\mathrm{H}[r]$ which creates the inequality in the first place, and $\mathrm{H}[r] = -\log \mathrm{p}(r)$ is minimized exactly when $\mathrm{p}(r)$ becomes maximal.

Hence, we choose the success probability ρ to do so: the maximum likelihood solution $\arg\max_p \mathrm{p}(r \mid \rho)$ is $\rho = \frac{r}{N}$. The Binomial distribution of R then has its mode, mean, and median at r.

Altogether, after substituting $\rho = \frac{r}{N}$ and rearranging, we see that the wanted approximation is actually an inequality:

$$\log \binom{N}{r} \leq -r \log \rho - (N - r) \log(1 - \rho) \tag{2.43}$$

$$= r \log \frac{N}{r} + (N - r) \log \frac{N}{N - r}. \tag{2.44}$$

Approximation Error $\mathrm{H}[r]$. The approximation error is just $\mathrm{H}[r]$ as we can read off from Equation 2.42. We can easily upper-bound it with $\mathrm{H}[r] \leq \log N$: First, $\mathrm{H}[R] \leq \log N$ as the uniform distribution with entropy $\log N$ is the maximum entropy distribution in this case (discrete random variable with finite support). Second, $\mathrm{H}[R]$ is the expectation over different $\mathrm{H}[R = r']$. We have chosen $\rho = \frac{r}{N}$ such that r is the mean of binomial distribution and has maximal probability mass. This means it has minimal information content. Hence, $\mathrm{H}[r] \leq \log N$ by contraposition as otherwise $\log N < \mathrm{H}[r] \leq \mathrm{H}[R]$.

Simplicity is the ultimate sophistication.

Leonardo da Vinci

3

Single Forward-Pass Aleatoric and Epistemic Uncertainty

In this chapter, we delve deeper into aleatoric and epistemic uncertainty and apply the insights gained to single forward-pass neural networks to disentangle these uncertainties.

Uncertainty quantification has garnered interest for such approaches because most well-known methods of uncertainty quantification in deep learning [Blundell et al., 2015; Gal and Ghahramani, 2016a; Lakshminarayanan et al., 2017; Wen et al., 2020; Dusenberry et al., 2020] require multiple forward passes at test time. Among these methods, deep ensembles have generally exhibited superior performance in uncertainty prediction [Ovadia et al., 2019]. However, their substantial memory and computational requirements during training and test time impede their adoption in real-life (e.g. mobile applications). As a result, there has been a growing interest in uncertainty quantification using deterministic single forward-pass neural networks, which offer a smaller footprint and reduced latency. We empirically validate our findings using active learning and out-of-distribution (OoD) detection on computer vision datasets.

To clarify, since OoD detection is not a well-defined term, we will investigate OoD detection for 'related distributions', such as CIFAR-10 versus SVHN or CIFAR-100, following the definition in Farquhar and Gal [2022]. Another term for this is 'near OoD' [Winkens et al., 2020]. Curiously, active learning and OoD detection are rarely evaluated together; thus, we will explore some of their nuances (recall §1.2.3) and contrast them.

Active Learning $\neq$ OoD Detection

We can conceptually differentiate active learning and OoD detection as follows:

- Active learning seeks to identify the most informative samples for labeling, while
- OoD detection aims to recognize samples that are not in-distribution, regardless of their informativeness.

This subtle yet crucial distinction implies that the informativeness signal in active learning will generally be useful for OoD detection (assuming the training data defines the in-distribution), *but not the other way around*: a sample that is 'obviously' OoD will not be informative for active learning. See Figure 3.1 for a visualization of this concept. In OoD literature, the terms 'near OoD' and 'far OoD' [Winkens et al., 2020] are often used to differentiate between points that are close to the in-distribution, leading to *conflicting* predictions, and those that are far away, making their detection *unanimous*.

Near-OoD points will likely be highly informative (high epistemic uncertainty) as the model might disagree about their interpretation, while far-OoD points will have low informativeness (low epistemic uncertainty) as the model will confidently detect

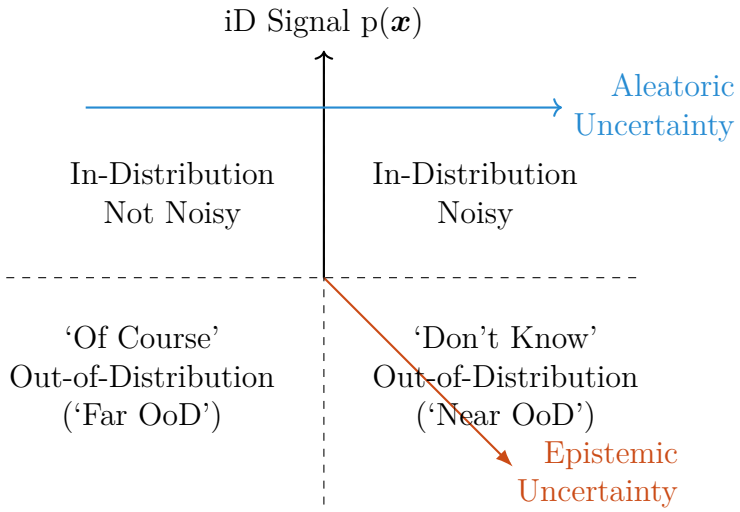


Figure 3.1: *The four quadrants of the OoD detection landscape.* Active learning focuses on the bottom-right quadrant, as data points in this area will be informative for the model. OoD detection is concerned with the lower half, as it aims to identify out-of-distribution data points (the training set is considered in-distribution). The 'Don't Know' quadrant is referred to as near OoD, and the 'Of Course' quadrant as far OoD. Consequently, feature-space density provides a suitable signal for active learning when the available pool data does not contain actual OoD data or outliers, as these will be confounded with informative in-distribution points. Equivalently, epistemic uncertainty as an active learning signal will only provide a reliable OoD signal for near OoD but not for far OoD.

them as OoD. This becomes an issue when using only epistemic uncertainty for OoD detection[1], as reported by Xia and Bouganis [2022]. Methods that employ outlier exposure are particularly prone to this problem, as epistemic uncertainty will also be low for outliers used during training. However, this issue can arise in general as the available training data increases: as the model parameters concentrate, epistemic uncertainty will decrease, and the model will become more confident in its predictions, causing more OoD data to become 'far OoD.'

As evident from this discussion, intriguing conceptual questions remain, yet active learning and OoD detection are rarely examined together. This chapter serves as an exception.

Relevant Literature

There are several single forward-pass uncertainty approaches in the literature that of particular relevance for this chapter. We will focus on methods that use feature-space distances and density [Settles, 2010; Lee et al., 2018b; van Amersfoort et al., 2020; Liu et al., 2020a; Postels et al., 2020].

Mahalanobis Distance. Among these approaches, Lee et al. [2018b] uses Mahalanobis distances to quantify uncertainty by fitting a class-wise Gaussian distribution (with shared covariance matrices) on the feature space of a pre-trained ResNet encoder. They do not consider the structure of the underlying feature-space however, which might

[1]It is unclear whether combining epistemic uncertainty with other signals has been explored in depth.

explain why their competitive results require input perturbations, the ensembling of OoD metrics over multiple layers, and fine-tuning on OoD hold-out data.

DUQ & SNGP. Two recent works in single forward-pass uncertainty, DUQ [van Amersfoort et al., 2020] and SNGP [Liu et al., 2020a], propose distance-aware output layers, in the form of RBFs (radial basis functions) or GPs (Gaussian processes), and introduce additional inductive biases in the feature extractor using a Jacobian penalty [Gulrajani et al., 2017] or spectral normalization [Miyato et al., 2018], respectively, which encourage smoothness and sensitivity in the latent space. These methods perform well and are almost competitive with deep ensembles on OoD benchmarks. However, they require training to be changed substantially, and introduce additional hyperparameters due to the specialized output layers used at training. Furthermore, DUQ and SNGP cannot disentangle aleatoric and epistemic uncertainty. Particularly, in DUQ, the feature representation of an ambiguous data point, high on aleatoric uncertainty, will be in between two centroids, but due to the exponential decay of the RBF it will seem far from both and thus have uncertainty similar to epistemically uncertain data points that are far from all centroids. In SNGP, the predictive variance is computed using a mean-field approximation of the softmax likelihood, which cannot be disentangled. The variance can also be computed using MC samples of the softmax likelihood which, in theory, can allow disentangling uncertainties (see Equation 1.34), but requires modelling the covariance between the classes, which is not the case in SNGP. We provide a more extensive review of related work in §3.6.

Outline

Concretely, we will focus on the following research questions:

1. Are complex methods to estimate uncertainty, like in DUQ and SNGP, necessary beyond feature-space regularization that encourages bi-Lipschitzness?
2. What are the conceptual challenges of different uncertainty metrics, and what instructive insights can be learned from these?
3. How can we disentangle aleatoric and epistemic uncertainty with single forward-pass neural networks, as DUQ and SNGP do not address this directly?

We make some simple but crucial observations that help answer our research questions in this chapter:

1. Entropy is arguably the wrong proxy for epistemic uncertainty, despite its frequent use for active learning and OoD detection. Specifically, we find that:
   (a) The predictive entropy as a metric confounds aleatoric and epistemic uncertainty (Figure 3.2(b)). This can be an issue in active learning in particular. Yet, this issue is often not visible for standard benchmark datasets without aleatoric noise. To examine this failure in more detail, we introduce a new dataset, Dirty-MNIST, which showcases the issue more clearly than artificially curated datasets like MNIST or CIFAR-10. *Dirty-MNIST* is an expanded version of MNIST [LeCun et al., 1998] with additional ambiguous digits (Ambiguous-MNIST) having multiple plausible labels and thus higher aleatoric uncertainty (Figure 3.2(a)).
   (b) The softmax entropy of a deterministic model trained with maximum likelihood, while being high for ambiguous points (i.e., with high aleatoric uncertainty), might not be consistent for points with high epistemic uncertainty,

i.e., the softmax entropy for an OoD sample might be low, high, or anything in between for different models trained on the same data (Figure 3.2(b)).

2. To disentangle aleatoric and epistemic uncertainty, feature-space density can be used to estimate epistemic uncertainty, and entropy for aleatoric uncertainty. However, feature-space regularization [Liu et al., 2020a] is crucial[2]. Without such regularization, feature-space density alone might not separate iD from OoD data, possibly explaining the limited empirical success of previous approaches which attempt to use feature-space density [Postels et al., 2020]. This can be seen in Figure 3.2(c) where the feature-space density of a VGG-16 or LeNet model is not able to differentiate iD Dirty-MNIST from OoD FashionMNIST, while a ResNet-18 with spectral normalization can do so better.
3. Objectives for density estimation and classification might have different optima (except on unambiguous, well-separable datasets), and using a single mixture model (e.g., a GMM) leads to suboptimal performance due to this *objective mismatch* [Murphy, 2012]. Hence, one should separately estimate the feature-space density for epistemic uncertainty and predictive entropy for aleatoric uncertainty.

Based on these observations, we examine an approach we call '*Deep Deterministic Uncertainty (DDU)*', which uses Gaussian Discriminant Analysis (GDA) for feature-space density on a trained model & the original softmax layer for estimating aleatoric uncertainty and making classification predictions. Using DDU, we empirically investigate whether complex methods to estimate uncertainty, like in DUQ and SNGP, are necessary beyond feature-space regularization that encourages bi-Lipschitzness. When we use spectral normalization like SNGP does, the short answer is an empirical no.

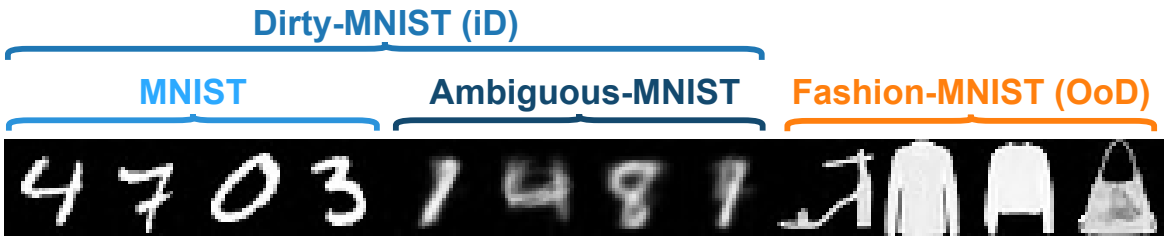
As we only perform GDA after training, the original softmax layer is trained using cross-entropy as a proper scoring rule [Gneiting and Raftery, 2007] and can be temperature-scaled to provide good in-distribution calibration and aleatoric uncertainty. DDU outperforms regular softmax neural networks, as illustrated in Figure 3.2. Furthermore, DDU is competitive with deep ensembles [Lakshminarayanan et al., 2017] and outperforms SNGP and DUQ [van Amersfoort et al., 2020; Liu et al., 2020a], with no changes to the model architecture beyond spectral normalization, in several OoD benchmarks and active learning settings. Using DeepLab-v3+ [Chen et al., 2017] on Pascal VOC 2012 [Everingham et al., 2010], we also show that DDU improves upon two classic uncertainty methods—MC Dropout [Gal and Ghahramani, 2016a] and deep ensembles—on the task of semantic segmentation, while being significantly faster to compute.

3.1 Entropy ≠ Epistemic Uncertainty

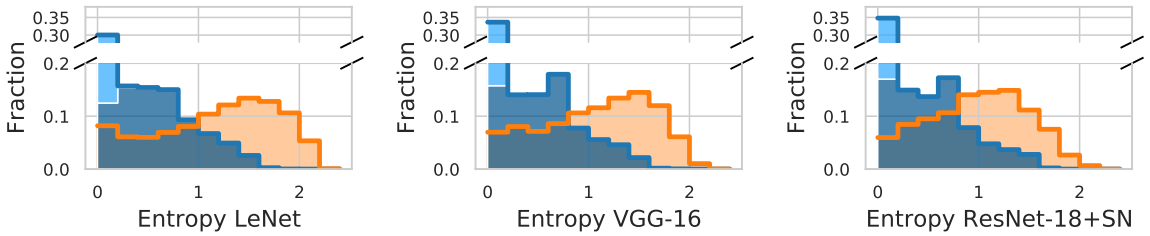
In this section, we observe that:

- using entropy for OoD detection is inherently problematic as it cannot distinguish between aleatoric uncertainty of ambiguous iD samples and the epistemic uncertainty of near OoD samples; and
- the softmax entropy of a single model is even more problematic as it is unreliable *specifically* for samples with high epistemic uncertainty, i.e., near OoD samples.

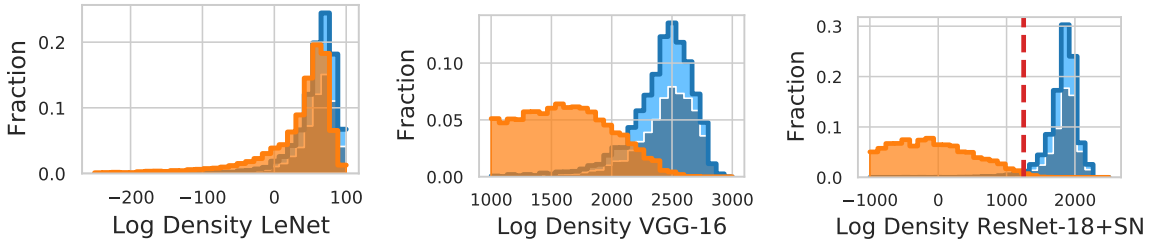
[2] [Pearce et al., 2021] argue for softmax confidence and entropy in their paper, yet feature-space density performs better in their experiments, too.



(a) Dirty-MNIST (iD, = MNIST + Ambiguous-MNIST) and FashionMNIST (OoD)



(b) Softmax entropy



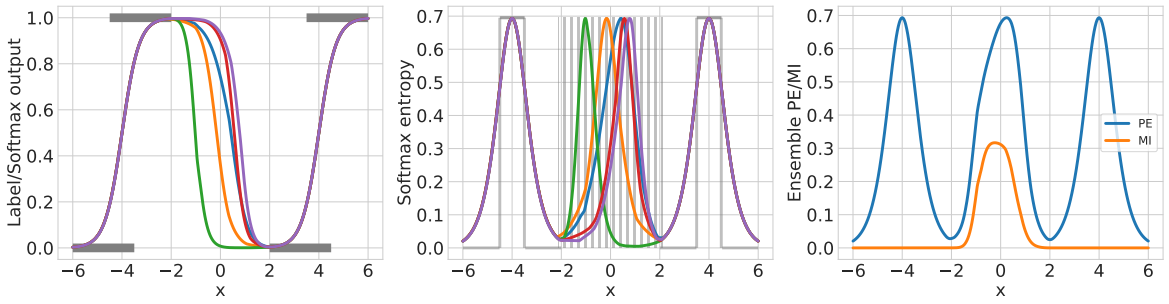
(c) Feature-space density

Figure 3.2: *Disentangling aleatoric and epistemic uncertainty on Dirty-MNIST (iD) and FashionMNIST (OoD)* ***(a)*** *requires using* softmax entropy ***(b)*** *and* feature-space density (GMM) ***(c)*** *with a well-regularized feature space (*ResNet-18+SN *vs* LeNet *&* VGG-16 *without smoothness & sensitivity).* **(b):** Softmax entropy captures aleatoric uncertainty for iD data (Dirty-MNIST), thereby separating unambiguous MNIST samples and Ambiguous-MNIST samples (stacked histogram). However, iD and OoD are confounded: softmax entropy has arbitrary values for OoD, indistinguishable from iD. **(c):** With a well-regularized feature space (DDU with ResNet-18+SN), iD and OoD densities do not overlap, capturing epistemic uncertainty. However, without such feature space (LeNet & VGG-16), feature density suffers from *feature collapse*: iD and OoD densities overlap. Generally, feature-space density confounds unambiguous and ambiguous iD samples as their densities overlap.

Both observations are tied to the very reason why a deep ensembles' mutual information captures epistemic uncertainty well and can be used to detect adversarial examples and near-OoD data, too [Smith and Gal, 2018]. To exemplify the issues, we will introduce *Dirty-MNIST* as a dataset with a long tail of ambiguous samples. We will conclude with an empirical analysis of the relationship between the softmax entropy and the predictive entropy (of a respective deep ensembles).

3.1.0.1 5-Ensemble Visualization

We start with a visualization of a 5-ensemble (with five deterministic softmax networks) to see how softmax entropy fails to capture epistemic uncertainty precisely because the mutual information (MI) of an ensemble does. This is illustrated in Figure 3.3 and provides intuition for the second point that softmax entropy is unreliable for samples



(a) Softmax Output **(b)** Softmax Entropy **(c)** 5-Ensemble

Figure 3.3: *Softmax outputs & entropies for 5 softmax models along with the predictive entropy (PE) and mutual information (MI) for the resulting 5-Ensemble.* (a) and (b) show that the softmax entropy is only reliably high for ambiguous iD points (±3.5–4.5), whereas it can be low or high for OoD points (-2–2). The different colors are the different ensemble components. Similarly, (c) shows that the MI of the ensemble is only high for OoD, whereas the PE is high for both OoD and for regions of ambiguity. See §3.1.0.1.

with high epistemic uncertainty. We train the networks on 1-dimensional data with binary labels 0 and 1. The data is shown in Figure 3.3(a). From Figure 3.3(a) and Figure 3.3(b), we find that the softmax entropy is high in regions of ambiguity where the label can be both 0 and 1 (i.e. x between -4.5 and -3.5, and between 3.5 and 4.5). This indicates that softmax entropy can capture aleatoric uncertainty. Furthermore, in the x interval $(-2, 2)$, we find that the deterministic softmax networks disagree in their predictions (see Figure 3.3(a)) and have softmax entropies which can be high, low or anywhere in between (see Figure 3.3(b)) following our claim in §3.2. In fact, this disagreement is the very reason why the MI of the ensemble is high in the interval $(-2, 2)$, thereby reliably capturing epistemic uncertainty. Finally, the predictive entropy (PE) of the ensemble is high both in the OoD interval $(-2, 2)$ as well as at points of ambiguity (i.e. at -4 and 4). This indicates that the PE of a deep ensemble captures both epistemic and aleatoric uncertainty well. From these visualizations, we draw the conclusion that the softmax entropy of a deterministic softmax model cannot capture epistemic uncertainty precisely because the MI of a deep ensemble can.

3.1.1 Dirty-MNIST

To show that entropy is inappropriate for OoD detection, we train a LeNet [LeCun et al., 1998], a VGG-16 [Simonyan and Zisserman, 2015] and a ResNet-18 with spectral normalization, ResNet+SN[3][He et al., 2016; Miyato et al., 2018] on *Dirty-MNIST*, a modified version of MNIST [LeCun et al., 1998] with additional ambiguous digits (Ambiguous-MNIST), depicted in Figure 3.2(a), which we introduce below.

Dirty-MNIST poses a challenge for using entropy for OoD detection as it confounds aleatoric and epistemic uncertainty: Figure 3.2(b) shows that the softmax entropy of a deterministic model is unable to distinguish between iD (Dirty-MNIST) and FashionMNIST samples [Xiao et al., 2017] as near OoD: the entropy for the latter heavily overlaps with the entropy for Ambiguous-MNIST samples. With the ambiguous data having various levels of aleatoric uncertainty, Dirty-MNIST is more representative

[3]Liu et al. [2020a] show that spectral normalization regularizes the latent space in a way that is beneficial for OoD detection, so we also include a model trained on this recent approach.

Figure 3.4: *Samples from Ambiguous-MNIST.*

of real-world datasets compared to well-cleaned curated datasets, like MNIST and CIFAR-10, commonly used for benchmarking [Krizhevsky, 2009].

3.1.1.1 Ambiguous-MNIST

Each sample in Ambiguous-MNIST is constructed by decoding a linear combination of latent representations of 2 different MNIST digits from a pre-trained VAE [Kingma and Welling, 2014]. Every decoded image is assigned several labels sampled from the softmax probabilities of an off-the-shelf MNIST neural network ensemble, with points filtered based on an ensemble's MI to remove "junk" images and then stratified class-wise based on their softmax entropy. All off-the-shelf MNIST neural networks were then discarded, and new models were trained to generate Figure 3.2—and as can be seen, the ambiguous points we generate indeed have high entropy regardless of the model architecture used. We create 60K such training and 10K test images to construct Ambiguous-MNIST. Finally, the Dirty-MNIST dataset in this experiment contains MNIST and Ambiguous-MNIST samples in a 1:1 ratio (thus, in total 120K training and 20K test samples). In Figure 3.4, we provide some samples from Ambiguous-MNIST. This provides intuition for the first point that entropy for OoD detection is inherently problematic as it cannot distinguish between aleatoric uncertainty of ambiguous iD samples and the epistemic uncertainty of near OoD samples.

3.1.2 Potential Pitfalls of Predictive and Softmax Entropy

Now let us discuss potential pitfalls of predictive entropy in general and softmax entropy of deterministic models in particular in more detail.

3.1.2.1 Potential Pitfalls of *Predictive Entropy*

Conceptually, *predictive entropy confounds epistemic and aleatoric uncertainty.* Since ensembling can also be interpreted as Bayesian Model Averaging [He et al., 2020; Wilson and Izmailov, 2020], with each ensemble member approximating a sample from a posterior, eq. (1.34) can be applied to ensembles to disentangle epistemic and aleatoric uncertainty. Both mutual information $\mathrm{I}[Y; \omega \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ and predictive entropy $\mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ could be used to detect OoD samples. However, previous empirical findings show predictive entropy outperforming mutual information [Malinin and Gales,

2018]. Indeed, much of the recent literature only focuses on predictive entropy for OoD detection. Table 3.1 shows a selection of recently published papers which use entropy or confidence as OoD score. Only two papers examine using mutual information with deep ensembles as OoD score at all. None of the papers examines the possible confounding of aleatoric and epistemic uncertainty when using predictive entropy or confidence, or the consistency issues of softmax entropy (and softmax confidence), detailed in §3.2. This list is not exhaustive, of course. We explain these findings using the following (obvious) observation:

Observation 3.1. When we *already know* that *either* aleatoric *or* epistemic uncertainty is *low* for an iD sample, predictive entropy is an appropriate measure of the other uncertainty type.

Thus, predictive entropy, as an upper-bound of mutual information, can separate iD and OoD data better when datasets are curated and have low aleatoric uncertainty. However, as seen in eq. (1.34), predictive entropy can be high for both iD ambiguous samples (high aleatoric) as well as near OoD samples (high epistemic) (see Figure 3.3) and might *not* be an effective measure for OoD detection when used with datasets that are not curated with ambiguous samples, like Dirty-MNIST, as seen in our active learning results.

3.1.2.2 Potential Pitfalls of *Softmax Entropy*

The softmax entropy for deterministic models trained with maximum likelihood can be inconsistent. As we have noted in §1.2.3, Equation 1.34 can be used with deep ensembles, as each ensemble member can be considered a sample from *some* distribution $\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$ over model parameters $\omega \subset \Omega$ (e.g. a uniform distribution over K trained ensemble members $\omega_1, ..., \omega_K$):

$$\underbrace{\mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]}_{\text{predictive}} = \underbrace{\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]}_{\text{epistemic}} + \underbrace{\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}]}_{\text{aleatoric (for iD } \boldsymbol{x})}. \tag{1.34}$$

Note that the mutual information $\mathrm{I}[Y; \omega \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ isolates epistemic from aleatoric uncertainty for deep ensembles as well, whereas the predictive entropy $\mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ (often used with deep ensembles) measures predictive uncertainty, which will be high whenever either epistemic or aleatoric uncertainties are high.

Rank Inconsistency. Crucially, the mechanism underlying deep ensemble uncertainty that can push epistemic uncertainty to be high on OoD data is the function disagreement between different ensemble members, i.e., arbitrary and disagreeing predictive extrapolations of the softmax models composing the ensemble due to a lack of relevant training data[4]: Deep ensembles demonstrating high epistemic uncertainty (mutual information) on OoD data entails that at least two ensemble members must *extrapolate differently* ('arbitrary extrapolations') on that data, since the predictive and aleatoric terms in Equation 1.34 must cancel out (leading the "aleatoric" term in eq. (1.34) to vanish [Smith and Gal, 2018]): This is because for OoD data points Equation 1.34 guarantees that whenever the epistemic uncertainty is high, the predictive entropy must be high as well. The following simple qualitative result captures this intuition:

[4]Something similar has also recently been reported in the context of ensemble calibration [Jordan, 2023].

Table 3.1: *A sample of recently published papers and OoD metrics.* Many recently published papers only use Predictive Entropy or Predictive Confidence (for deep ensembles) or Softmax Confidence (for deterministic models) as OoD scores without addressing the possible confounding of aleatoric and epistemic uncertainty, that is ambiguous iD samples with OoD samples. Only two papers examine using mutual information with deep ensembles as OoD score at all.

Title [Citation]	Softmax Confidence	Predictive Confidence	Predictive Entropy	Mutual Information
A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks [Hendrycks and Gimpel, 2017]	✓	✗	✗	✗
Deep Anomaly Detection with Outlier Exposure [Hendrycks et al., 2019]	✓	✗	✗	✗
Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks [Liang et al., 2018]	✓	✗	✗	✗
Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples [Lee et al., 2018a]	✓	✗	✗	✗
Learning Confidence for Out-of-Distribution Detection in Neural Networks [DeVries and Taylor, 2018]	✓	✗	✗	✗
Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles [Lakshminarayanan et al., 2017]	✗	✓	✓	✗
Predictive Uncertainty Estimation via Prior Networks [Malinin and Gales, 2018]	✗	✓	✓	✓
Ensemble Distribution Distillation [Malinin et al., 2019]	✗	✗	✓	✓
Generalized ODIN: Detecting Out-of-Distribution Image Without Learning From Out-of-Distribution Data [Hsu et al., 2020]	✓	✗	✗	✗
Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks [Kristiadi et al., 2020]	✗	✓	✗	✗

Proposition 3.1. *Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be points such that $\boldsymbol{x}_1$ has **higher** epistemic uncertainty than $\boldsymbol{x}_2$ under the ensemble:*

$$\mathrm{I}[Y_1; \omega \mid x_1, \mathcal{D}^{\text{train}}] > \mathrm{I}[Y_2; \omega \mid x_2, \mathcal{D}^{\text{train}}] + \delta, \tag{3.1}$$

$\delta \geq 0$. Further, assume both have similar predictive entropy

$$|\, \mathrm{H}[Y_1 \mid x_1, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y_2 \mid x_2, \mathcal{D}^{\text{train}}]| \leq \epsilon, \tag{3.2}$$

*$\epsilon \geq 0$. Then, there exist sets of ensemble members Ω' (assuming a countable ensemble), with $\mathrm{p}(\Omega' \mid \mathcal{D}^{\text{train}}) > 0$, such that for all softmax models $\omega' \in \Omega'$ the softmax entropy of $\boldsymbol{x}_1$ is **lower** than the softmax entropy of $\boldsymbol{x}_2$:*

$$\mathrm{H}[Y_1 \mid \boldsymbol{x}_1, \omega'] < \mathrm{H}[Y_2 \mid \boldsymbol{x}_2, \omega'] - (\delta - \epsilon). \tag{3.3}$$

Proof. From Equation 1.34, we obtain:

$$\begin{aligned} &|\, \mathrm{H}[Y_1 \mid x_1, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y_2 \mid x_2, \mathcal{D}^{\text{train}}]| \leq \epsilon \\ \Leftrightarrow &|\, \mathrm{I}[Y_1; \omega \mid x_1, \mathcal{D}^{\text{train}}] + \mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y_1 \mid x_1, \omega]] \end{aligned} \tag{3.4}$$

$$- \mathrm{I}[Y_2; \omega \mid x_2, \mathcal{D}^{\text{train}}] - \mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y_2 \mid x_2, \omega]]| \leq \epsilon, \tag{3.5}$$

and hence we have:

$$\begin{aligned} &\mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y_1 \mid x_1, \omega]] - \mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y_2 \mid x_2, \omega]] \\ &+ \underbrace{(\mathrm{I}[Y_1; \omega \mid x_1, \mathcal{D}^{\text{train}}] - \mathrm{I}[Y_2; \omega \mid x_2, \mathcal{D}^{\text{train}}])}_{>\delta} \leq \epsilon. \end{aligned} \tag{3.6}$$

We can rearrange the terms:

$$\mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y_1 \mid x_1, \omega]] < \mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y_2 \mid x_2, \omega]] - (\delta - \epsilon). \tag{3.7}$$

Now, the statement follows by contraposition: if $\mathrm{H}[Y_1 \mid \boldsymbol{x}_1, \omega] \geq \mathrm{H}[Y_2 \mid \boldsymbol{x}_2, \omega] - (\delta - \epsilon)$ for all ω, the monotonicity of the expectation would yield $\mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y_1 \mid \boldsymbol{x}_1, \omega]] \geq \mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y_2 \mid \boldsymbol{x}_2, \omega]] - (\delta - \epsilon)$. Thus, there is a set Ω' with $\mathrm{p}(\Omega' \mid \mathcal{D}^{\text{train}}) > 0$, such that

$$\mathrm{H}[Y_1 \mid x_1, \omega] < \mathrm{H}[Y_2 \mid x_2, \omega] - (\delta - \epsilon), \tag{3.8}$$

for all $\omega \in \Omega'$. □

If a sample is assigned higher epistemic uncertainty (in the form of mutual information) by a deep ensemble than another sample, it will necessarily be assigned lower softmax entropy by at least one of the ensemble's members. As a result, a priori, we cannot know whether a softmax model preserves the order or not, and *the empirical observation that the mutual information of an ensemble can quantify epistemic uncertainty well implies that the softmax entropy of a deterministic model might not.* We see this in Figure 3.2(b), 3.3 and in §3.1.3 where softmax entropy for OoD samples can be high, low or anywhere in between. While *not all* model architectures might behave like this, when the mutual information of a deep ensemble works well empirically, Proposition 3.1 holds.

This directly impacts the quality of the OoD detection, as we will verify in §3.1.3 and §3.5. For OoD detection, the changes in the score ranks can induce additional false positive or false negative—the AUROC, for example, directly measures the probability that an iD point has higher score than an OoD point. For active learning, the changes in the order can lead to less uninformative samples being selected—on the other hand, the additional noise this introduces can also be beneficial as we investigate in §5.

Bias-Variance Trade-Off. We can take a different perspective and view the softmax entropy of a single model as a (biased) estimator of the predictive entropy of the ensemble. What is the root mean squared error of this estimator?

Proposition 3.2. *The root mean squared error of the softmax entropy of a single model as an estimator of the predictive entropy of the ensemble:*

$$\mathrm{RMSE}_{\boldsymbol{\omega}}(\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}], \mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]) = \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})}[\big(\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}] - \mathrm{H}[Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]\big)^2]^{1/2} \tag{3.9}$$

decomposes into a bias-variance trade-off with bias $\mathrm{I}[Y; \boldsymbol{\omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$ *and variance* $\mathrm{Var}_{\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}]]$.

Proof. We will drop conditioning on $\mathcal{D}^{\text{train}}$ for this proof.

1. We use that $\mathrm{I}[Y; \boldsymbol{\omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] \triangleq \mathrm{H}[Y \mid \boldsymbol{x}] - \mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}, \mathcal{D}^{\text{train}}]$ (from §2),

2. As $\mathrm{H}[Y \mid \boldsymbol{x}]$ is independent of $\boldsymbol{\Omega}$, we have:

$$\mathrm{Var}_{\boldsymbol{\omega}}[\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}]] = \mathrm{Var}_{\boldsymbol{\omega}}[\mathrm{I}[Y; \boldsymbol{\omega} \mid \boldsymbol{x}]]. \tag{3.10}$$

3. Expanding the variance, we obtain:

$$\begin{aligned} \mathrm{Var}_{\boldsymbol{\omega}}[\mathrm{I}[Y; \boldsymbol{\omega} \mid \boldsymbol{x}]] &= \mathbb{E}_{\boldsymbol{\omega}}[\mathrm{I}[Y; \boldsymbol{\omega} \mid \boldsymbol{x}]^2] - \mathbb{E}_{\boldsymbol{\omega}}[\mathrm{I}[Y; \boldsymbol{\omega} \mid \boldsymbol{x}]]^2 && (3.11) \\ &= \mathbb{E}_{\boldsymbol{\omega}}[\mathrm{I}[Y; \boldsymbol{\omega} \mid \boldsymbol{x}]^2] - \mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}]^2. && (3.12) \end{aligned}$$

4. We substitute the definition of $\mathrm{I}[Y; \boldsymbol{\omega} \mid \boldsymbol{x}]$ in the first term and use the equality of the variances above. Rearranging:

$$\mathbb{E}_{\boldsymbol{\omega}}[(\mathrm{H}[Y \mid \boldsymbol{x}] - \mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}, \mathcal{D}^{\text{train}}])^2] = \mathrm{Var}_{\boldsymbol{\omega}}[\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}]] + \mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}]^2. \tag{3.13}$$

5. Taking the square root yields the result:

$$\begin{aligned} &\mathrm{RMSE}_{\boldsymbol{\omega}}(\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\Omega}], \mathrm{H}[Y \mid \boldsymbol{x}]) && (3.14) \\ &\quad = \mathbb{E}_{\boldsymbol{\omega}}[(\mathrm{H}[Y \mid \boldsymbol{x}] - \mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}, \mathcal{D}^{\text{train}}])^2]^1/2 && (3.15) \\ &\quad = \sqrt{\underbrace{\mathrm{Var}_{\boldsymbol{\omega}}[\mathrm{H}[Y \mid \boldsymbol{x}, \boldsymbol{\omega}]]}_{\text{Variance}} + \underbrace{\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}]^2}_{\text{Bias}}}. && (3.16) \end{aligned}$$

□

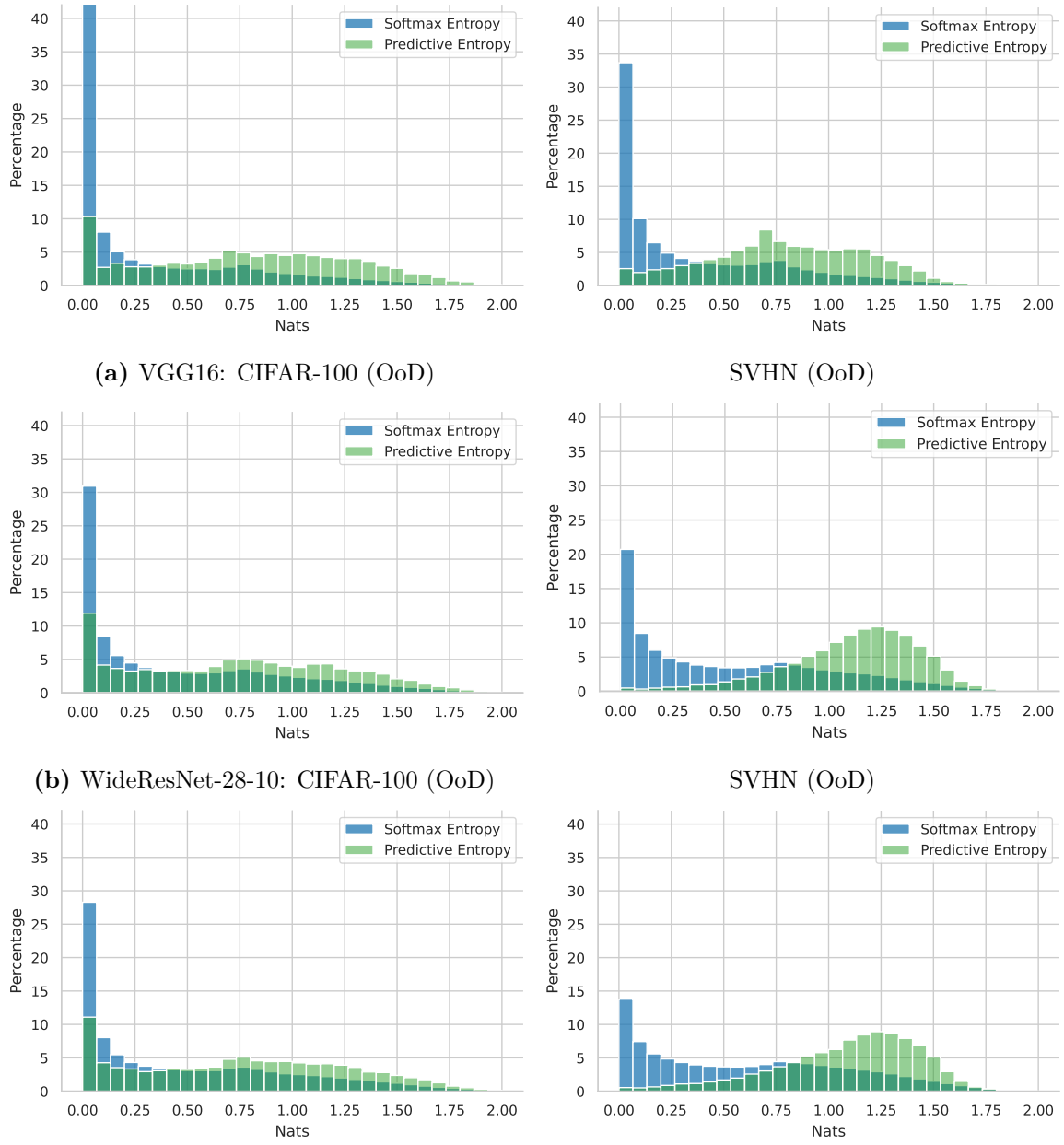
Hence, the expected deviation from the predictive entropy becomes the largest for high mutual information/epistemic uncertainty. This is in line with the empirical observations that the softmax entropy of a deterministic model might not be a good estimator of the predictive entropy of the ensemble from §3.1.3 below.

Given that we can view an ensemble member as a single deterministic model and vice versa, these two propositions provide an intuitive explanation for why single deterministic models can report inconsistent and widely varying predictive entropies and confidence scores for OoD samples for which a deep ensemble would report high epistemic uncertainty (expected information gain) and high predictive entropy.

3.1.3 Qualitative Empirical Validation

While the 5-ensemble visualization and qualitative and quantitative statements provide us with an intuition for why ensemble members and thus deterministic models cannot provide epistemic uncertainty reliably through their softmax entropies, to gain further insights, we empirically analyze the relationship between softmax entropies and predictive entropies more precisely next.

To do so we train deep ensembles of 25 members on CIFAR-10 using different model architectures (VGG-16, Wide-ResNet-28-10/+SN) and visualize the relationship between the softmax entropies of the ensemble members, and the mutual information (epistemic uncertainty/EIG/BALD) and the predictive entropies of the overall ensemble on two OoD datasets (CIFAR-100 and SVHN). Details for the experiment setup can be found in §3.5.



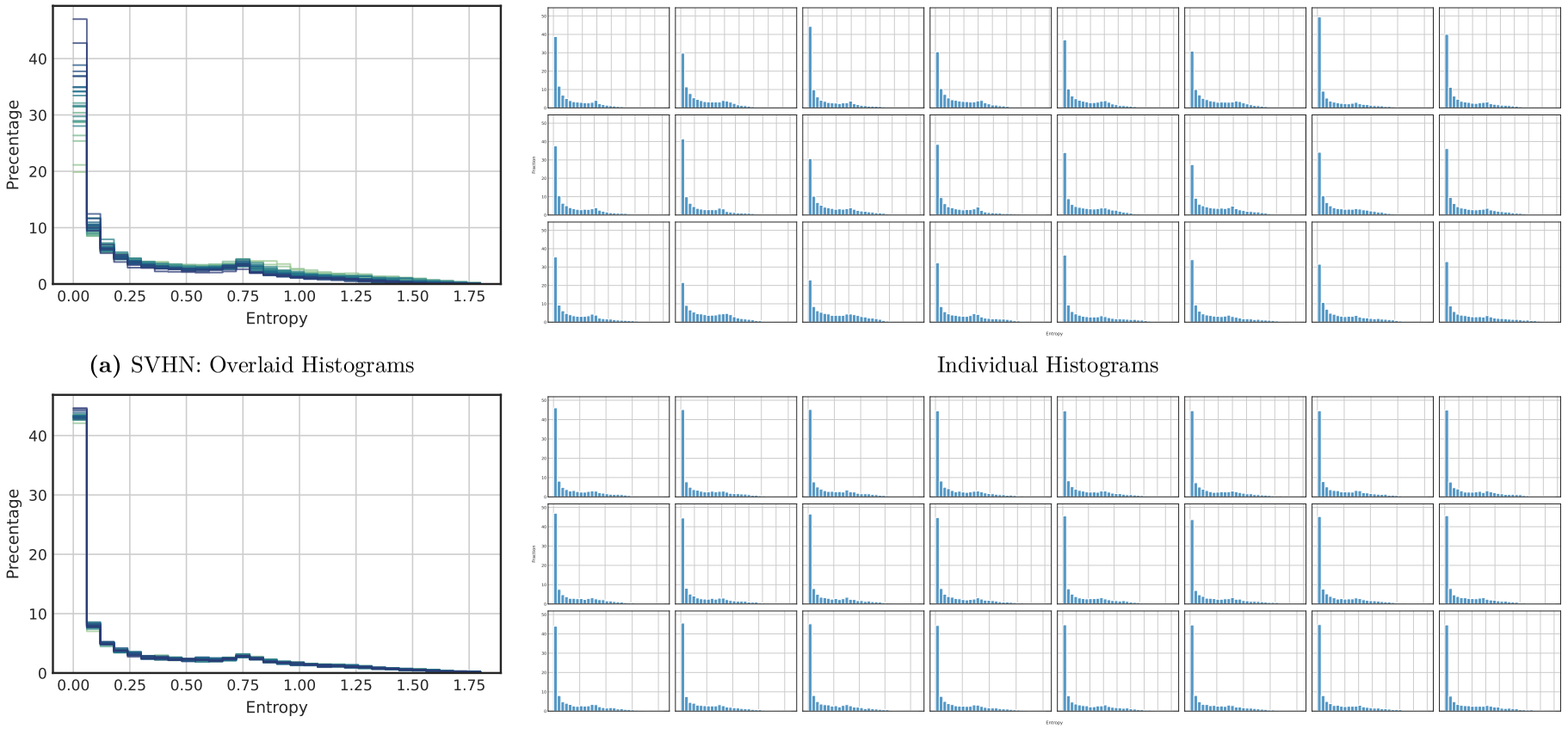
(a) VGG16: CIFAR-100 (OoD) SVHN (OoD)

(b) WideResNet-28-10: CIFAR-100 (OoD) SVHN (OoD)

(c) WideResNet-28-10+SN: CIFAR-100 (OoD) SVHN (OoD)

Figure 3.5: *Softmax entropy vs. predictive entropy trained on CIFAR-10 (iD) using different model architectures (25 models each).* We see that predictive and softmax entropy are distributed very differently. (`+SN` refers to models trained with spectral norm and small modifications to the architecture described in §3.4.1.)

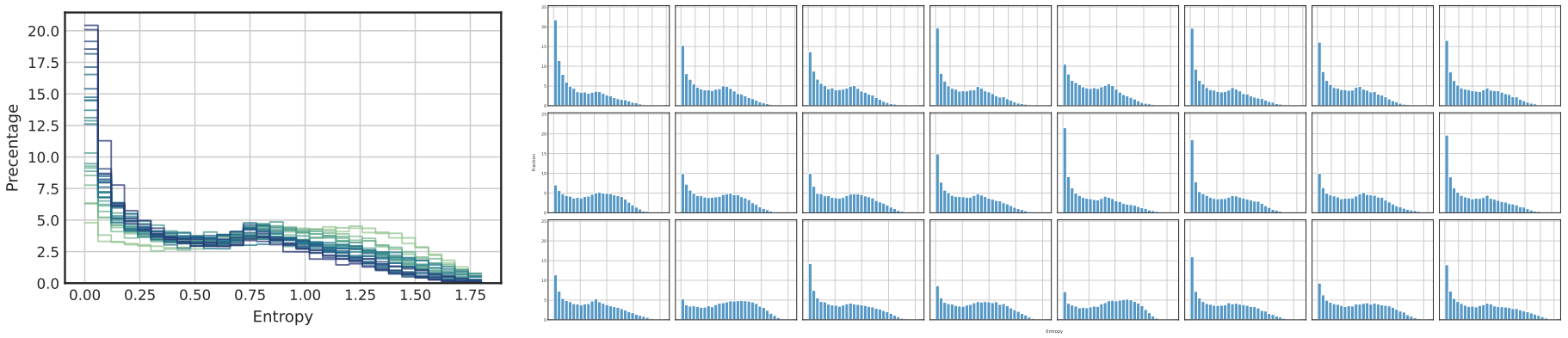
Softmax Entropy vs Predictive Entropy. Figure 3.5 shows that the softmax entropy of the ensemble members is not a good estimator of the predictive entropy of the ensemble. Not even the histograms look similar. Figures 3.6 and 3.7 show histograms of the different ensemble members. We see that on SVHN there is a lot more variation in the softmax entropy distribution of the ensemble members than on CIFAR-100. This is also reflected in the predictive entropy of the ensemble, which is often higher on SVHN than on CIFAR-100. Figure 3.8, which also includes the mutual information



(a) SVHN: Overlaid Histograms

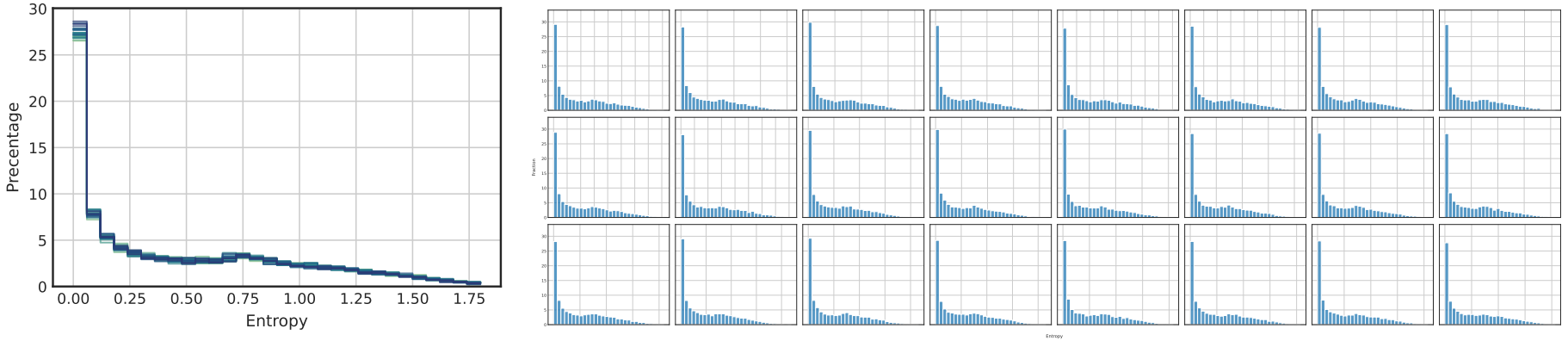
(b) CIFAR-100: Overlaid Histograms

Figure 3.6: *VGG-16: Empirical distribution of softmax entropies of the ensemble members (25 models, 24 are shown).* We see that the empirical distribution of softmax entropies can vary a lot between different models of the same architecture. **(a)** the empirical distribution of softmax entropies for SVHN as OoD displays a lot of variance; while **(b)** the empirical distribution of softmax entropies for CIFAR-100 as OoD display very little variation. `+SN` refers to models trained with spectral norm and small modifications to the architecture described in §3.4.1.



(a) SVHN: Overlaid Histograms

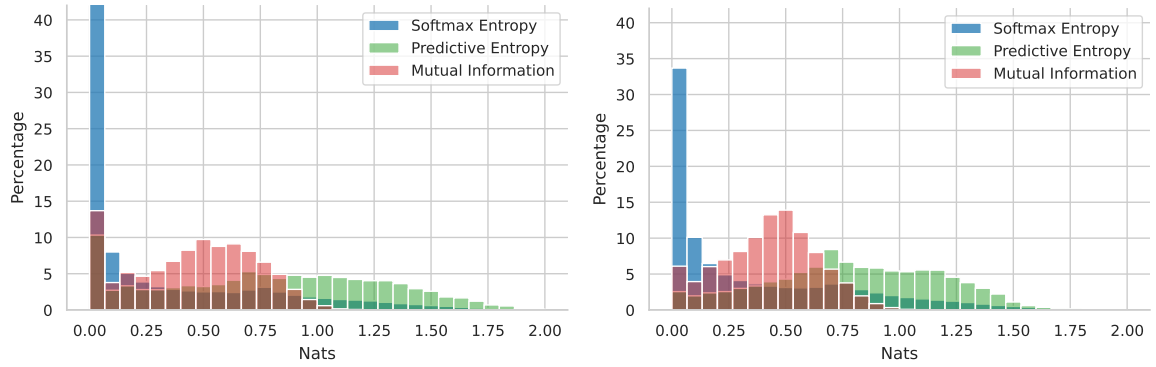
Individual Histograms



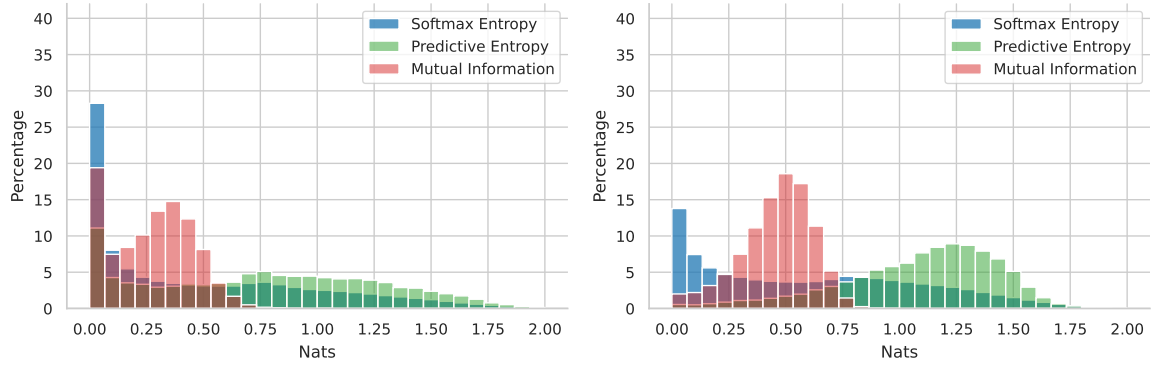
(b) CIFAR-100: Overlaid Histograms

Individual Histograms

Figure 3.7: *Wide-ResNet-28-10+SN: Empirical distribution of softmax entropies of the ensemble members (25 models, 24 are shown).* We see that the empirical distribution of softmax entropies can vary a lot between different models of the same architecture. **(a)** the empirical distribution of softmax entropies for SVHN as OoD displays a lot of variation between models; while **(b)** the empirical distribution of softmax entropies for CIFAR-100 as OoD display very little variation. (`+SN` refers to models trained with spectral norm and small modifications to the architecture described in §3.4.1.)



(a) VGG16: CIFAR-100 (OoD) SVHN (OoD)



(b) WideResNet-28-10+SN: CIFAR-100 (OoD) SVHN (OoD)

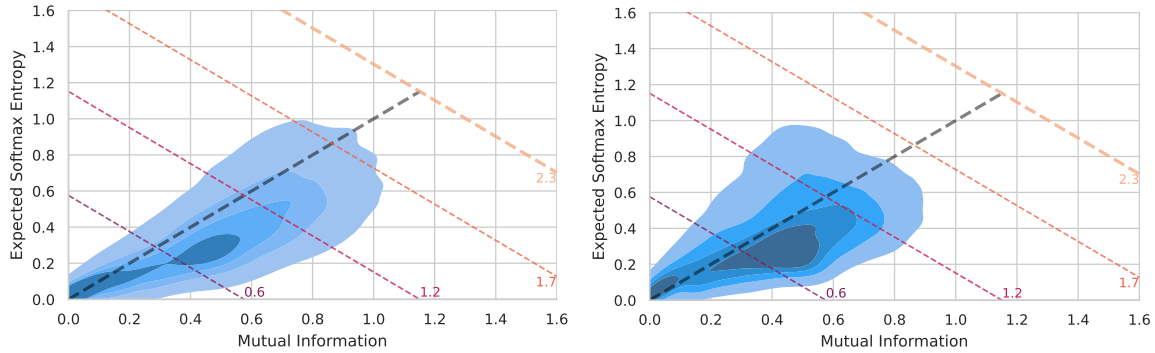
Figure 3.8: *Mutual Information (Epistemic Uncertainty).* Trained on CIFAR-10 (iD) using different model architectures (25 models each). The mutual information is better behaved than the softmax entropies, but less broad than the predictive entropy. (`+SN` refers to models trained with spectral norm and small modifications to the architecture described in §3.4.1.)

(EIG) of the ensemble, validates this: the mutual information near 0 is much lower on CIFAR-100 than on SVHN. The model expresses less disagreement on CIFAR-100.

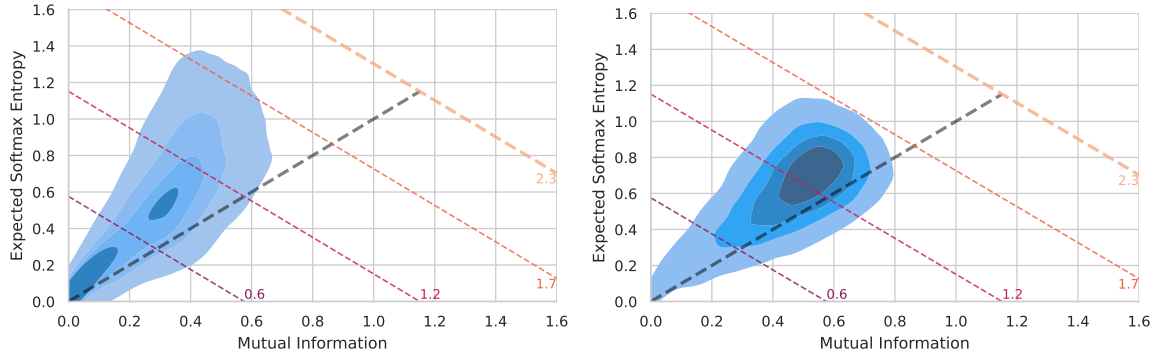
Figure 3.9 provides a more complete picture of all three information quantities we care about (mutual information, predictive entropy, and expected softmax entropy). The model architectures perform very differently on the two datasets: the VGG-16 model has smaller predictive entropies overall which pushes the expected softmax entropies below the mutual information for many OoD samples.

Figure 3.10 shows density plots for predictive entropy versus softmax entropy (and mutual information versus softmax entropy) across all ensemble members. We see that most of the softmax entropies are very low even when the same sample has very high predictive entropy or mutual information. This implies that even small aleatoric uncertainty in iD samples will lead the model to confound them with OoD samples.

Softmax Entropy Variance. Figure 3.11 show density plots for the mutual information vs the variance of the softmax entropy of the ensemble. As the mutual information increases, the variance of the softmax entropy first increases and then decreases slightly. Overall, it is mostly quite low compared to the mutual information. Figure 3.12 validates this. It shows a histogram of the RMSE following Proposition 3.2 vs the mutual information. Indeed, the error is dominated by its bias, the mutual information.



(a) VGG16: CIFAR-100 (OoD) SVHN (OoD)

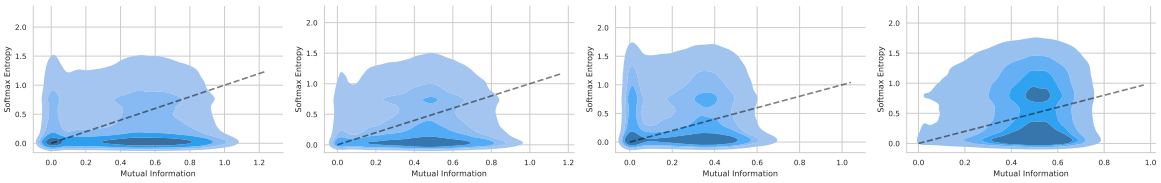


(b) WideResNet-28-10+SN: CIFAR-100 (OoD) SVHN (OoD)

Figure 3.9: *Mutual Information (Epistemic Uncertainty) vs Expected Softmax Entropy (Aleatoric Uncertainty) vs Predictive Entropy (Total Uncertainty).* Trained on CIFAR-10 (iD) using different model architectures (25 models each). The predictive entropy is shown via its iso-lines (anti-diagonals). Darker is denser.(`+SN` refers to models trained with spectral norm and small modifications to the architecture described in §3.4.1.)

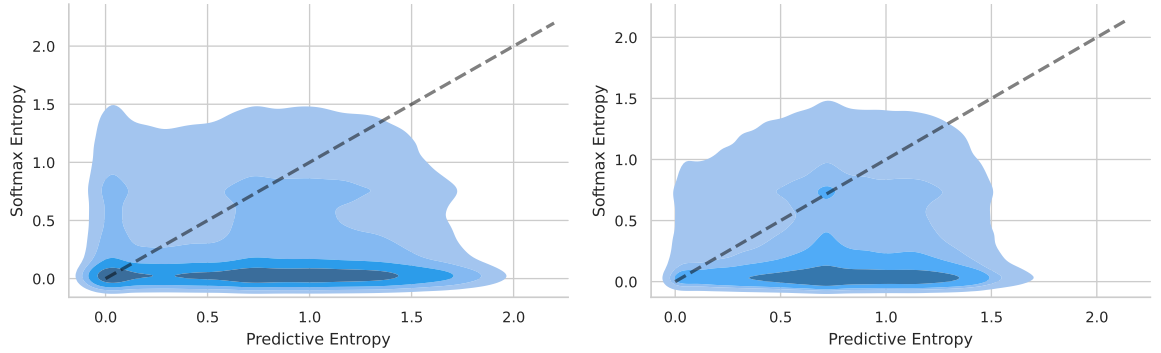
3.1.4 Discussion

Both through quantitative and qualitative statements as well as through empirical validation, we see that neither the predictive entropy of deep ensembles nor the softmax entropy of deterministic models is appropriate for measuring epistemic uncertainty and OoD detection tasks. This holds in particularly for real world datasets that contain more ambiguous data than the curated datasets that are employed for benchmarking. Proposition 3.1 shows that the softmax entropy does not provide a stable ranking of points with high epistemic uncertainty, and that indeed, the predictive entropy of deep ensembles captures epistemic uncertainty through the variance of the softmax entropies of deterministic models. Proposition 3.2 shows that when we view softmax entropies as estimators of the respective predictive entropies of an ensemble and decompose the error using a bias-variance trade-off, the mutual information (epistemic uncertainty) is the bias of this estimator, and the variance of the softmax entropy is the variance of the estimator. Additionally, we have empirically analyzed several model architectures and found that the softmax entropy varies considerably across the ensemble members and is neither a good measure of epistemic uncertainty (with the mutual information as proxy) nor of predictive entropy, which could be surprising given that we usually do not differentiate between softmax and predictive entropy.

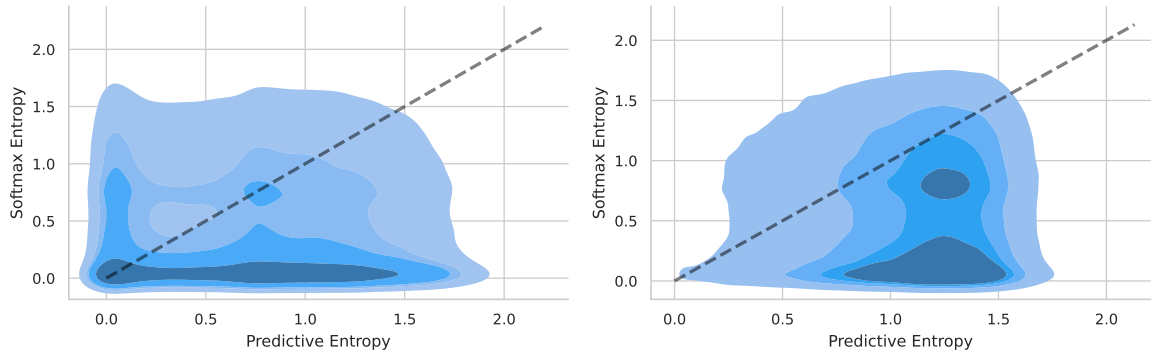


VGG16: CIFAR-100 SVHN WRN+SN: CIFAR-100 SVHN

(a) Mutual Information vs Softmax Entropy



VGG16: CIFAR-100 (OoD) SVHN (OoD)



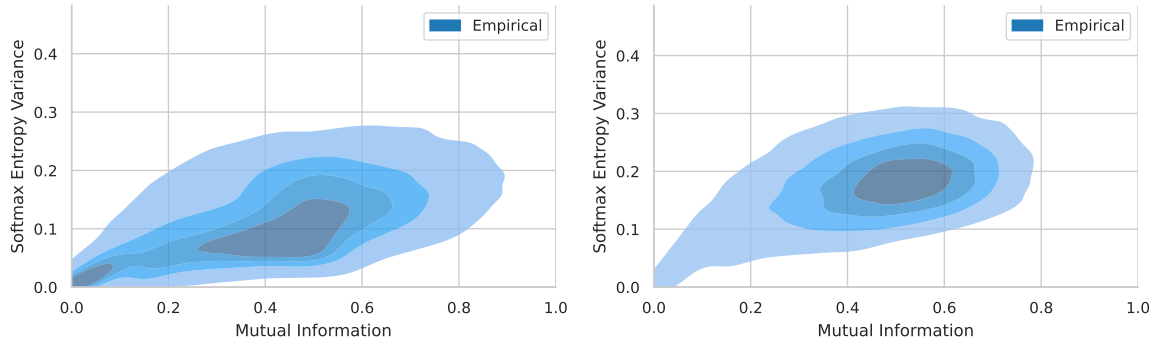
WideResNet-28-10+SN: CIFAR-100 (OoD) SVHN (OoD)

(b) Predictive Entropy vs Softmax Entropy

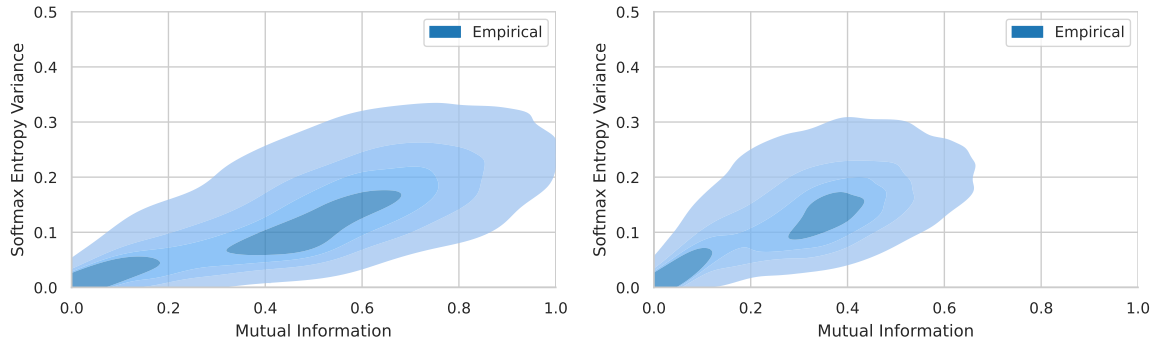
Figure 3.10: *Predictive Entropy (Total Uncertainty) & Mutual Information (Epistemic Uncertainty) vs Expected Softmax Entropy (Aleatoric Uncertainty)* Trained on CIFAR-10 (iD) using different model architectures (25 models each). Darker is denser.(`+SN` refers to models trained with spectral norm and small modifications to the architecture described in §3.4.1.)

While our criticism of softmax entropy seems generally valid, mutual information (expected information gain/epistemic uncertainty) is not necessarily a good measure for *far OoD detection* as we have argued in §3. Predictive entropy can be high for both far OoD points and near OoD points, which it can confound with ambiguous iD points, however.

Is there a way to ensure that the model will have high uncertainty for OoD points in general (and thus minimize the amount of possible far OoD points)? Yes, feature-space regularization, whether implicit and explicit, can do just that: in §1.2.3, we introduced bi-Lipschitzness as a concept that can encourage OoD points to be separated from iD points, thus allowing different ensemble members to potentially (hopefully) express higher disagreement on these points.



(a) SVHN (OoD): VGG-16 Wide-ResNet-28-10+SN



(b) CIFAR-100 (OoD): VGG-16 Wide-ResNet-28-10+SN

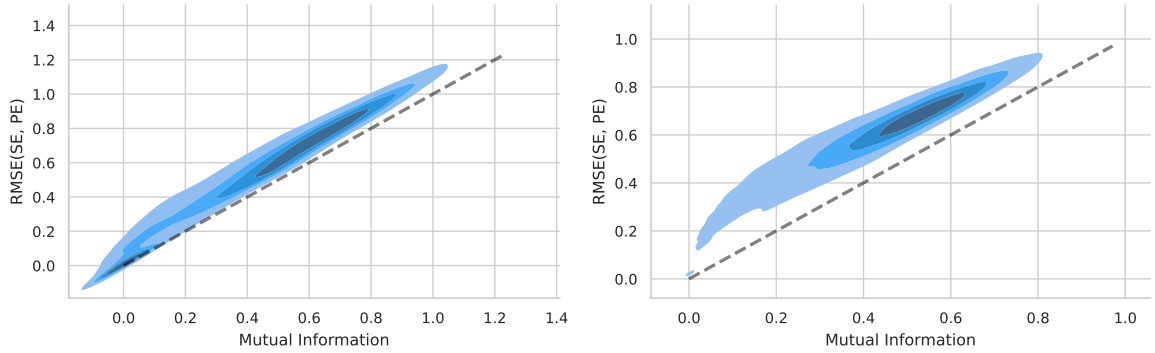
Figure 3.11: *Mutual Information (Epistemic Uncertainty) vs Softmax Entropy Variance (ESV).* Trained on CIFAR-10 (iD) using different model architectures (25 models each). For both SVHN and CIFAR-100 as OoD dataset, we see that the softmax variance first increases as the mutual information increases and then decreases. Overall, compared to the mutual information, it is quite small. (`+SN` refers to models trained with spectral norm and small modifications to the architecture described in §3.4.1).

Importantly, note that outlier exposure, a popular method to improve OoD detection performance, trains the model on held-out "OoD" data. This breaks the equivalence "*OoD data* $\iff$ *high epistemic uncertainty*" that underlies using epistemic uncertainty for OoD detection: training using outlier exposure transforms epistemic uncertainty into aleatoric uncertainty, which can be captured by the softmax entropy of deterministic models as well as the predictive entropy of a deep ensemble—*even though it confounds ambiguous iD samples with OoD samples.* However, is such data still truly OoD when we start training or fine-tuning on it, or are we simply moving the goal posts?

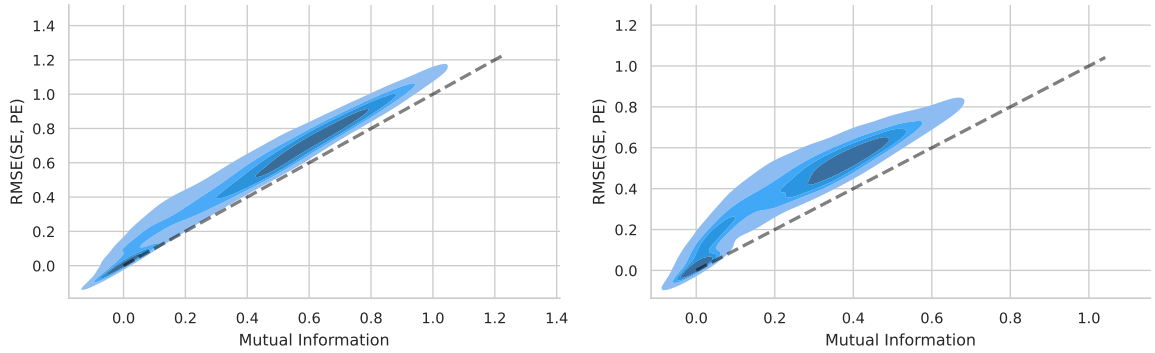
3.2 Aleatoric & Epistemic Uncertainty

The failure of softmax entropy to capture epistemic uncertainty motivates us to study feature-space density as an alternative for single-forward pass approaches. Indeed, feature-space density is a well-known acquisition function in active learning [Settles, 2010].

Epistemic Uncertainty via Feature-Space Density. In §1.2.3, we noted that feature-space density conceptually fulfills the requirement of epistemic uncertainty as being a reducible uncertainty. Here, we will investigate this further—or rather, to



(a) SVHN (OoD): VGG-16 Wide-ResNet-28-10+SN



(b) CIFAR-100 (OoD): VGG-16 Wide-ResNet-28-10+SN

Figure 3.12: *Mutual Information (Epistemic Uncertainty) vs (Entropy) Root Mean Squared Error (RMSE, Proposition 3.2).* Trained on CIFAR-10 (iD) using different model architectures (25 models each). For both SVHN and CIFAR-100 as OoD dataset, we see that the RMSE increases as the mutual information (the bias) increases. The bias seems to be dominating the error compared to the softmax variance. (`+SN` refers to models trained with spectral norm and small modifications to the architecture described in §3.4.1).

be more precise—we will investigate the *negative* feature-space density as a proxy for epistemic uncertainty.

With a well-regularized feature space using spectral normalization, we find that simply performing GDA (Gaussian Discriminant Analysis) *after training* as feature-space density estimator can reliably capture epistemic uncertainty. However, unlike Lee et al. [2018b], which does not place any constraints on the feature space, training on "OoD" hold-out data, feature ensembling, and input pre-processing are not needed to obtain good performance (see Table 3.5). This results in a conceptually simpler method. Moreover, we find that using a separate covariance matrix for each class improves OoD detection performance as compared to a shared covariance matrix.

Crucially, feature-space density cannot express aleatoric uncertainty: an iD sample ought to have a high density regardless of whether it is ambiguous (with high aleatoric uncertainty) or not, as is the case with Dirty-MNIST in Figure 3.2(c). However, softmax entropy is suitable for that. Hence, we can use the softmax entropy to predict the aleatoric uncertainty on iD samples with low epistemic uncertainty, for which is it meaningfully defined:

Observation 3.2. The softmax entropy of a deterministic model together with its feature-space density can disentangle epistemic and aleatoric uncertainty while either alone cannot.

Sensitivity & Smoothness. As discussed in §1.2.3, feature extractors that do not fulfill a sensitivity constraint can suffer from feature collapse: they might map OoD samples to iD regions of the feature space. Thus, we encourage these properties using residual connections and spectral normalization. The effects of these properties on feature collapse are visible in Figure 3.2. In the case of feature collapse, we must have *some* OoD inputs for which the features are mapped on top of the features of iD inputs. The distances of these OoD features to each class centroid must be equal to the distances of the corresponding iD inputs to class centroids, and hence the density for these OoD inputs must be equal to the density of the iD inputs. If the density histograms for given iD and OoD samples do not overlap, no feature collapse can be present for those samples. We see no overlapping densities in Figure 3.2(c)(right most), therefore we indeed have no feature collapse between Dirty-MNIST and FashionMNIST. Similarly, smoothness constraints are necessary to encourage generalization when using feature-space density as a proxy for epistemic uncertainty [van Amersfoort et al., 2020].

Gaussian & Linear Discriminant Analysis. Given feature vector $\boldsymbol{z}$ and class label y, we model the class-conditional probabilities $\mathrm{q}(\boldsymbol{z} \mid y)$ of feature vectors given class labels using Gaussian Discriminant Analysis (GDA), which is a generative classifier $\mathrm{q}(\boldsymbol{z}, y)$ based on a Gaussian mixture model (GMM) $\mathrm{q}(\boldsymbol{z} \mid y)\ \mathrm{q}(y)$, where $\mathrm{q}(y)$ is the class prior and $\mathrm{q}(\boldsymbol{z} \mid y)$ is a multivariate Gaussian density. GDA is closely related to Linear Discriminant Analysis (LDA) [Murphy, 2012]. To predict the class label $\mathrm{q}(y \mid \boldsymbol{z})$ of a new sample $\boldsymbol{z}$, we use Bayes' theorem:

$$\mathrm{q}(y \mid \boldsymbol{z}) \propto \mathrm{q}(\boldsymbol{z} \mid y)\ \mathrm{q}(y) \tag{3.17}$$

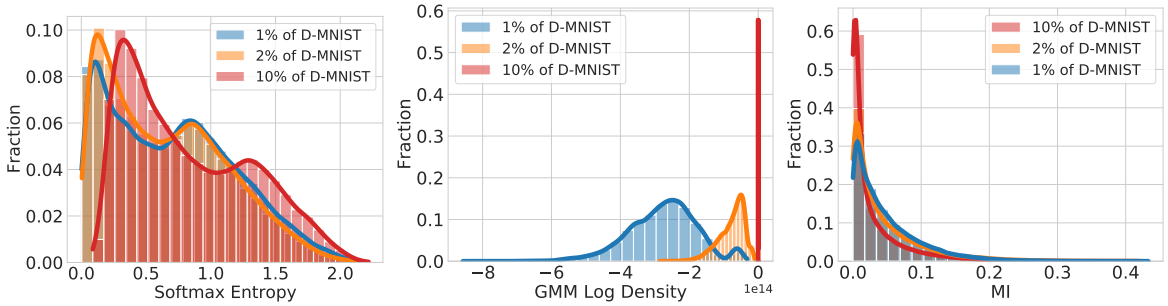
In GDA, each class is modeled by a Gaussian mixture component with its own covariance matrix. The class-conditional probabilities are given by $\mathrm{q}(\boldsymbol{z}|y = c) = \mathcal{N}(\boldsymbol{z};\, \mu_c,\, \Sigma_c)$, where μ_c and Σ_c are the mean and covariance matrix for class c, respectively.

On the other hand, LDA models the class-conditional probabilities using a single multivariate Gaussian distribution per class, with a *shared* covariance matrix Σ among all classes. The class-conditional probabilities in LDA are given by $\mathrm{q}(\boldsymbol{z} \mid y = c) \sim \mathcal{N}(\boldsymbol{z};\, \mu_c,\, \Sigma)$.

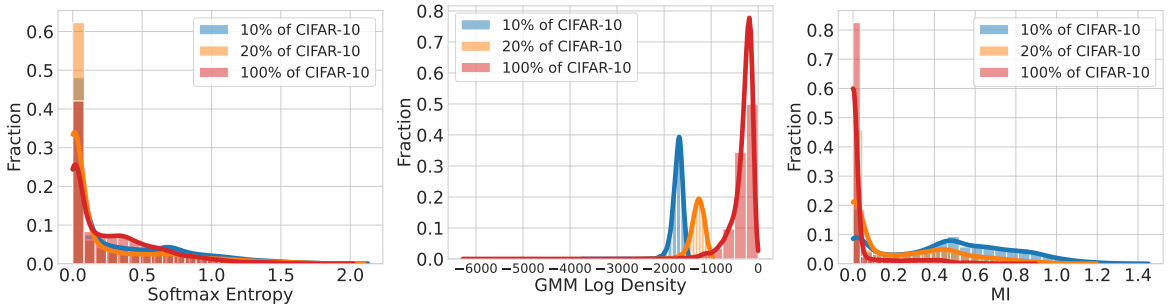
Thus, the difference between GDA and LDA lies in their modeling of the covariance matrix. While GDA allows for a separate covariance matrix for each class, LDA assumes a shared covariance matrix among all classes. This distinction influences the flexibility and complexity of the models: GDA is more flexible but potentially more prone to overfitting, while LDA is more constrained but may be more robust in some scenarios.

Scope. We only use GDA for estimating the feature-space density as it is straightforward to implement and does not require performing expectation maximization or variational inference like other density estimators. Normalizing flows [Dinh et al., 2015] or other more complex density estimators might provide even better density estimates, of course. Yet despite its simplicity, GDA is already sufficient to outperform other more complex approaches and obtain good results as we report in §3.5.

Our focus is on obtaining a well-regularized feature space using spectral normalization in model architectures with residual connections, following Liu et al. [2020a]. Note



(a) Dirty-MNIST



(b) CIFAR-10

Figure 3.13: *Comparison of epistemic and aleatoric uncertainty captured by ResNet-18+SN on increasingly large subsets of Dirty-MNIST and CIFAR-10. Feature density captures epistemic uncertainty which reduces when the model is trained on increasingly large subsets of training data, whereas softmax entropy (SE) does not. For comparison, we also plot a deep-ensemble's epistemic uncertainty, through mutual information (MI) for the same settings. For more details, see Table 3.2.*

that unsupervised methods using contrastive learning [Winkens et al., 2020] might also obtain such a feature space by training on very large datasets, but training on them can be very expensive [Sun et al., 2017]. Generally, as the amount of training data available grows and feature extractors improve, the quality of feature representations might improve as well. The underlying motivation of this chapter is that simple approaches will remain more applicable than more complex ones as our empirical results suggest.

3.2.1 Reducible Feature-Space Density & Irreducible Entropy

To empirically verify the connection between feature-space density and epistemic uncertainty on the one hand and the connection between softmax entropy and aleatoric uncertainty on the other hand, we train ResNet-18+SN models on increasingly large subsets of Dirty-MNIST and CIFAR-10 and evaluate the epistemic and aleatoric uncertainty on the corresponding test sets using the feature-space density and softmax entropy, respectively. Moreover, we also train a 5-ensemble on the same subsets of data and use the ensemble's mutual information as a baseline measure of epistemic uncertainty.

In Figure 3.13 and Table 3.2, we see that with larger training sets, the average feature-space density increases which is consistent with the epistemic uncertainty decreasing as more data is available as reducible uncertainty. This is also evident from the consistent strong positive correlation between the negative log density and mutual information of the ensemble. On the other hand, the softmax entropy stays roughly the same which is consistent with aleatoric uncertainty as irreducible uncertainty, which

Table 3.2: *Average softmax entropy (SE) and feature-space density of the test set for models trained on different amounts of the training set (Dirty-MNIST and CIFAR-10) behave consistently with aleatoric and epistemic uncertainty.* Aleatoric uncertainty for individual samples does not change much as more data is added to the training set while epistemic uncertainty decreases as more data is added. This is also consistent with Table 3 in Kendall and Gal [2017]. Finally, we observe a consistent strong positive correlation between the negative log feature space density and the mutual information (MI) of a deep ensemble trained on the same subsets of data for both Dirty-MNIST and CIFAR-10. However, the correlation between softmax entropy and MI is not consistent.

Training Set	Avg Test SE (≈)	Avg Test Log GMM Density (↑)	Avg Test MI	Correlation(SE || MI)	Correlation(-Log GMM Density || MI)
1% of D-MNIST	0.7407	$-2.7268e+14$	0.0476		
2% of D-MNIST	0.6580	$-7.8633e+13$	0.0447	−0.79897	0.8132
10% of D-MNIST	0.8295	−1279.1753	0.0286		
10% of CIFAR-10	0.3189	−1715.3516	0.4573		
20% of CIFAR-10	0.2305	−1290.1726	0.2247	0.5663	0.9556
100% of CIFAR-10	0.2747	−324.8040	0.0479		

is independent of the training data. Importantly, all of this is also consistent with the experiments comparing epistemic and aleatoric uncertainty on increasing training set sizes in Table 3 of [Kendall and Gal, 2017].

3.3 Objective Mismatch

So far, we have seen that the feature-space density of a model can be used as a proxy to quantify the epistemic uncertainty of the model, and that the softmax entropy of the model can be used as a proxy to quantify the aleatoric uncertainty of the model.

Why did we use softmax entropy to estimate aleatoric uncertainty? Why not use the predictive probability $\mathrm{q}(y \mid \boldsymbol{z})$ of the GDA model that we use to estimate the feature-space density $\mathrm{p}(\boldsymbol{z})$? It is not a matter of convenience but rather a matter of potentially conflicting objectives:

In this section, we show that the feature-space density and softmax entropy are optimal for the respective uncertainty quantification tasks, and that this is because of an *objective mismatch* between the two tasks. The predictive probability induced by a feature-density estimator will generally not be well-calibrated as there is an objective mismatch. This was overlooked in previous research on uncertainty quantification for deterministic models: Lee et al. [2018b]; Liu et al. [2020a]; van Amersfoort et al. [2020]; He et al. [2016]; Postels et al. [2020]. Specifically, a mixture model $\mathrm{q}(y, \boldsymbol{z}) = \sum_y \mathrm{q}(\boldsymbol{z} \mid y)\,\mathrm{q}(y)$, using one component per class, cannot be optimal for both feature-space density and predictive distribution estimation as there is an *objective mismatch* [Murphy, 2012, Ex. 4.20, p. 145]:

Proposition 3.3. *For an input $\boldsymbol{x}$, let $\boldsymbol{z} = f_\theta(\boldsymbol{x})$ denote its feature representation in a feature extractor f_θ with parameters θ. Then the following hold:*

1. *a discriminative classifier $\mathrm{q}_\theta(y \mid \boldsymbol{z})$, e.g. a softmax layer, is well-calibrated in its predictions when it maximizes the conditional log-likelihood $\mathrm{q}_\theta(y \mid \boldsymbol{z})$;*
2. *a feature-space density estimator $\mathrm{q}_\theta(\boldsymbol{z})$ is optimal when it maximizes the marginalized log-likelihood $\log \mathrm{q}(\boldsymbol{z})$;*
3. *a mixture model $\mathrm{q}_\theta(y, \boldsymbol{z}) = \sum_y \mathrm{q}_\theta(\boldsymbol{z} \mid y)\,\mathrm{q}(y)$ might not be able to maximize both objectives, conditional log-likelihood and marginalized log-likelihood, at the same time.*

*In the specific instance that a GMM with one component per class does maximize both, the resulting model must be a GDA (*but the opposite does not hold*).*

Following the notation form §1.2.1, $\mathrm{q}_\theta(\cdot)$ denotes a probability distribution parameterized by θ. As cross-entropies upper-bound the respective entropies, we have: $\mathrm{H}_\theta[Y, Z] \geq \mathrm{H}[Y, Z], \mathrm{H}_\theta[Z] \geq \mathrm{H}[Z]$, and $\mathrm{H}_\theta[Y \mid Z] \geq \mathrm{H}[Y \mid Z]$.

Proof. We prove the statements in order. The first two are trivial.
1. The conditional log-likelihood is a strictly proper scoring rule [Gneiting and Raftery, 2007]. The optimization objective can be rewritten as

$$\max_\theta \mathbb{E}[\log \mathrm{p}_\theta(y \mid z)] = -\min_\theta \mathrm{H}_\theta[Y \mid Z] \leq -\mathrm{H}[Y \mid Z]. \tag{3.18}$$

An optimal discriminative classifier $\mathrm{q}_\theta(y \mid \boldsymbol{z})$ with equality above would thus capture the true (empirical) distribution everywhere: $\mathrm{q}_\theta(y \mid \boldsymbol{z}) = \hat{\mathrm{p}}_{\text{true}}(y \mid \boldsymbol{z})$.
2. For density estimation $\mathrm{q}_\theta(\boldsymbol{z})$, we maximize the log-likelihood $\mathbb{E}[\log \mathrm{q}_\theta(\boldsymbol{z})]$ using the empirical data distribution. We can rewrite this as

$$\max_\theta \mathbb{E}_{\hat{\mathrm{p}}_{\text{true}}(y,z)} \log \mathrm{p}_\theta(z) = -\min_\theta \mathrm{H}_\theta[Z] \leq -\mathrm{H}[Z]. \tag{3.19}$$

When we have equality, we have $\mathrm{p}_\theta(\boldsymbol{z}) = \hat{\mathrm{p}}_{\text{true}}(\boldsymbol{z})$.
3. Using $\mathrm{H}_\theta[Y, Z] = \mathrm{H}_\theta[Y \mid Z] + \mathrm{H}_\theta[Z]$, we can relate the objectives from Equation 3.18 and (3.19) to each other. First, we characterize a shared optimum, and then we show that both objectives are generally not minimized at the same time. For both objectives to be minimized, we have $\nabla \mathrm{H}_\theta[Y \mid Z] = 0$ and $\nabla \mathrm{H}_\theta[Z] = 0$, and we obtain

$$\nabla \mathrm{H}_\theta[Y, Z] = \nabla \mathrm{H}_\theta[Y \mid Z] + \nabla \mathrm{H}_\theta[Z] = 0. \tag{3.20}$$

From this we conclude that minimizing both objectives also minimizes $\mathrm{H}_\theta[Y, Z]$, and that generally the objectives trade-off with each other at stationary points θ of $\mathrm{H}_\theta[Y, Z]$:

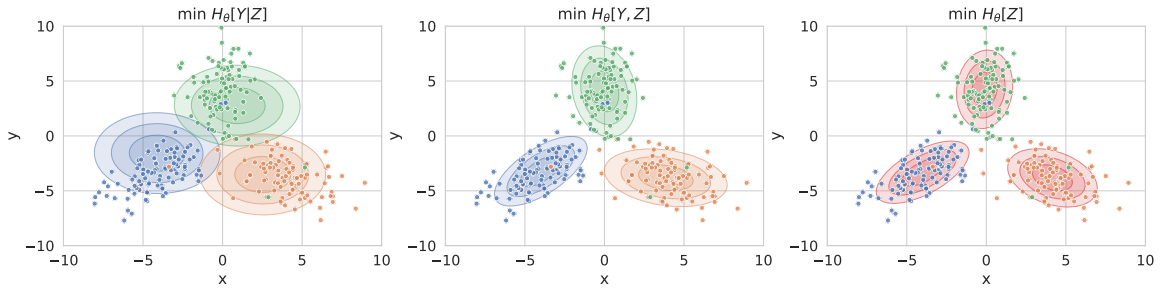
$$\nabla \mathrm{H}_\theta[Y \mid Z] = -\nabla \mathrm{H}_\theta[Z] \quad \text{when } \nabla \mathrm{H}_\theta[Y, Z] = 0. \tag{3.21}$$

This tells us that to construct a case where the optima do not coincide, discriminative classification needs to be opposed better density estimation.

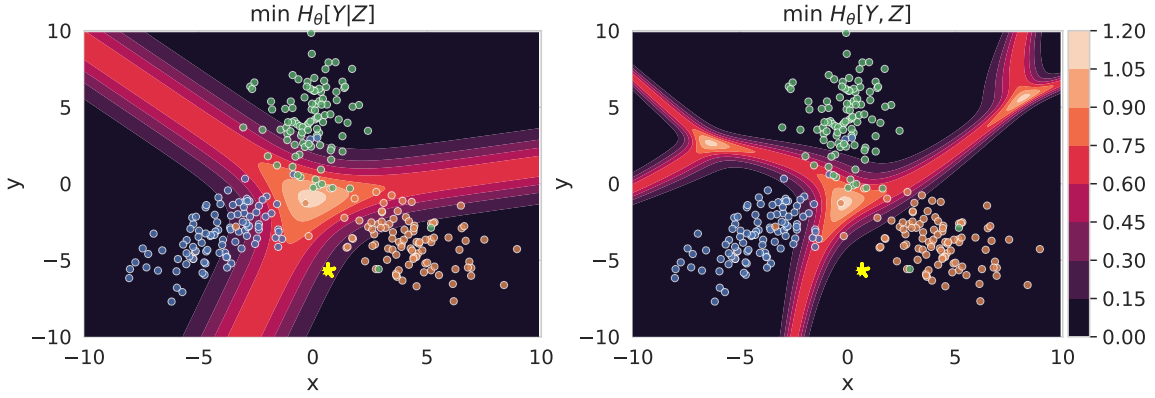
□

Existence. Let us also show that the cases described above can occur. Specifically, when we have a GMM with one component per class, minimizing $\mathrm{H}_\theta[Y, Z]$ on an empirical data distribution is equivalent to Gaussian Discriminant Analysis, as is easy to check, and minimizing $\mathrm{H}_\theta[Z]$ is equivalent to fitting a density estimator, following Equation 3.19. The difference is that using a GMM as a density estimator does not constrain the component assignment for a sample, unlike in GDA. Overall, we see that *both objectives can be minimized at the same time exactly when the feature representations of different classes are perfectly separated*, such that a GMM fit as density estimator would assign each class's feature representations to a single component.

By the above, we can construct simple examples for both cases: if the samples of different classes are not separated in feature-space, optima for the objectives will not coincide. For example, if samples were drawn from the same Gaussian and labeled



(a) *Density.* Contours at 68.26%, 95.44%, and 99.7%.



(b) *Entropy.* Darker is lower.

Figure 3.14: *3-component GMM fitted to a synthetic dataset with 3 different classes (differently colored) with 4% label noise using different objectives.* **(a):** The optima for conditional log-likelihood $\mathrm{H}_\theta[Y \mid Z]$, joint log-likelihood $\mathrm{H}_\theta[Y, Z]$, and marginalized log-likelihood $\mathrm{H}_\theta[Z]$ all differ. Hence, the best calibrated model ($\mathrm{H}_\theta[Y \mid Z]$) will not provide the best density estimate ($\mathrm{H}_\theta[Z]$), and vice-versa. **(b):** A mixture model that optimizes $\mathrm{H}_\theta[Y, Z]$ (GDA) does not have calibrated decision boundaries for aleatoric uncertainty: the ambiguous sample (due to label noise) marked by the yellow star has no aleatoric uncertainty under the GDA model. See §3.3.1 for details.

randomly. On the other hand, when the features of different classes all lie in well-separated clusters, GDA can minimize all objectives at the same time.

Given that perfect separation is impossible with ambiguous data for a GMM, a shared optimum will be rare with noisy real-world data, but only then would GDA be optimal. In all other cases, GDA does not optimize both objectives, and neither can any other GMM with one component per class.

Equation 3.21 tells us that a GMM fit using Expectation Maximization is likely a better density estimator than GDA, and a softmax layer is a better classifier, as optimizing the density objective $\mathrm{H}[Z]$ or softmax objective $\mathrm{H}_\theta[Y \mid Z]$ using gradient descent will move away from the optimum of the GDA objective

3.3.1 Toy Example

To explain Proposition 3.3 in an intuitive way, we focus on a simple synthetic 2D dataset with three classes and 4% label noise and fit GMMs using the different objectives. To construct, the dataset we sample "latents" $\boldsymbol{z}$ from three different Gaussians (each representing a different class y) with 4% label noise. Following the construction in the proof, this will lead the objectives to have different optima. Note that label noise and

Table 3.3: *Realized objective scores (columns) for different optimization objectives (rows) for the synthetic 2D toy example depicted in Figure 3.14.* Smaller is better. We see that each objective minimizes its own score while being suboptimal in regard to the other two objectives (when it is possible to compute the scores). This empirically further validates Proposition 3.3.

Loss → / Objective ↓	$\mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Y} \mid \boldsymbol{Z}]$ (↓)	$\mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Y}, \boldsymbol{Z}]$ (↓)	$\mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Z}]$ (↓)
$\min \mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Y} \mid \boldsymbol{Z}]$	**0.1794**	5.4924	5.2995
$\min \mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Y}, \boldsymbol{Z}]$	0.2165	**4.9744**	4.7580
$\min \mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Z}]$	n/a	n/a	**4.7073**

non-separability of features are common issues in real-world datasets.

In Table 3.3 and Figure 3.14(a), we see that each solution minimizes its own objective best. The regular GMM (which optimizes the density) provides the best density model (best fit according to the entropy), while the LDA (like a softmax linear layer) provides the best NLL for the labels. The GDA provides a density model that is almost as good as the GMM. Let us discuss the different objectives in Figure 3.14 and the resulting scores in more detail:

$\min \mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Y} \mid \boldsymbol{Z}]$. A softmax linear layer is equivalent to an LDA (Linear Discriminant Analysis) with conditional likelihood as detailed in Murphy [2012]. We optimize an LDA with the usual objective "$\min -1/N \sum \log \mathrm{q}(y \mid \boldsymbol{z})$", i.e. the cross-entropy of $\mathrm{q}(y \mid \boldsymbol{z})$ or (average) negative log-likelihood (NLL). Because we optimize only $\mathrm{q}(y \mid \boldsymbol{z})$, $\mathrm{q}(\boldsymbol{z})$ does not affect the objective and is thus not optimized. Indeed, the components do not actually cover the latents well, as can be seen in the first density plot of Figure 3.14(a). However, it does provide the lowest NLL.

$\min \mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Y}, \boldsymbol{Z}]$. We optimize a GDA for the combined objective "$\min -1/N \sum \log \mathrm{q}(y, \boldsymbol{z})$", i.e. the cross-entropy of $\mathrm{q}(y, \boldsymbol{z})$. We use the shorthand "$\min \mathrm{H}_{\theta}[Y \mid Z]$" for this.

$\min \mathbf{H}_{\boldsymbol{\theta}}[\boldsymbol{Z}]$. We optimize a GMM for the objective "$\min -1/N \sum \log \mathrm{q}(\boldsymbol{z})$", i.e. the cross-entropy of $\mathrm{q}(\boldsymbol{z})$. We use the shorthand "$\min \mathrm{H}_{\theta}[Z]$" for this. Scores for $\mathrm{H}_{\theta}[Y \mid Z]$ and $\mathrm{H}_{\theta}[Y, Z]$ for the third objective $\min \mathrm{H}_{\theta}[Z]$ are not provided in Table 3.3 as it does not depend on Y, and hence the different components do not actually model the different classes. Hence, we also use a single color to visualize the components for this objective in Figure 3.14(a).

Entropy Plot. Looking at the entropy plots in Figure 3.14(b), we first notice that the LDA solution optimized for $\min \mathrm{H}_{\theta}[Y \mid Z]$ has a wide decision boundary. This is due to the overlap of the Gaussian components, which is necessary to provide the right aleatoric uncertainty.

Optimizing the negative log-likelihood $-\log \mathrm{p}(y \mid \boldsymbol{z})$ is a proper scoring rule, and hence is optimized for calibrated predictions.

Compared to this, the GDA solution (optimized for $\min \mathrm{H}_{\theta}[Y, Z]$) has a much narrower decision boundary and cannot capture aleatoric uncertainty as well. This is reflected in the higher NLL. Moreover, unlike for LDA, GDA decision boundaries behave differently than one would naively expect due to the untied covariance matrices. They can be curved, and the decisions change far away from the data [Murphy, 2012].

To show the difference between the two objectives we have marked an ambiguous point near $(0, -5)$ with a yellow star. Under the first objective $\min \mathrm{H}_\theta[Y, Z]$, the point has high aleatoric uncertainty (high entropy), as seen in the left entropy plot while under the second objective ($\min \mathrm{H}_\theta[Y, Z]$) the point is only assigned very low entropy. The GDA optimized for the second objective thus is overconfident.

As explained above, we do not show an entropy plot of $Y \mid Z$ for the third objective $\min \mathrm{H}_\theta[Z]$ in Figure 3.14(b) because the objective does not depend on Y, and there are thus no class predictions.

Intuitively, for aleatoric uncertainty, the Gaussian components need to overlap to express high aleatoric uncertainty (uncertain labelling). At the same time, this necessarily provides looser density estimates. On the other hand, the GDA density is much tighter, but this comes at the cost of NLL for classification because it cannot express aleatoric uncertainty that well. Figure 3.14 visualizes how the objectives trade-off between each other, and why we use the softmax layer trained for $\mathrm{p}(y \mid \boldsymbol{z})$ for classification and aleatoric uncertainty, and GDA as density model for $\mathrm{q}(\boldsymbol{z})$.

3.3.2 Discussion

We have shown that the objectives for a GMM with one component per class are obviously not equivalent, and that the optima of these objectives do not need to coincide. Hence, importantly, the above statement tells us that we can expect better performance by *using both a discriminative classifier (e.g., softmax layer) to capture aleatoric uncertainty for iD samples and a separate feature-density estimator to capture epistemic uncertainty even on a model trained using conditional log-likelihood, i.e. the usual cross-entropy objective.* As noted in the previous section §3.2, we focus on the GDA objective instead of the GMM objective as it is easier to compute, even though it also suffers from an objective mismatch. However, both in our toy example (Table 3.3), where the difference between the GDA objective to the softmax objective is larger than the difference to the GMM objective for the relevant metrics, and in our experiments (Table 3.9), we found that the GDA objective is sufficient for good performance.

3.4 Deep Deterministic Uncertainty

Based on the previous sections, we propose the following method:

Deep Deterministic Uncertainty (DDU) is a simple baseline method for uncertainty quantification for deterministic neural networks. It uses *a deterministic neural network with an appropriately regularized feature-space, using spectral normalization [Liu et al., 2020a]*, which can disentangle aleatoric and epistemic uncertainty. It estimates:

1. aleatoric uncertainty using softmax entropy, and
2. epistemic uncertainty by fitting a GDA after training.

There is no need for any additional precessing steps: no hold-out "OoD" data, feature ensembling, or input pre-processing, unlike in Lee et al. [2018b].

Ensuring Sensitivity & Smoothness. We ensure sensitivity and smoothness using spectral normalization in models with residual connections. In addition, we make minor changes to the standard residual block to further encourage sensitivity without sacrificing accuracy (see details in §3.4.1).

Algorithm 1 Deep Deterministic Uncertainty

Definitions:
- Regularized feature extractor $f_\theta : \boldsymbol{x} \to \mathbb{R}^d$
- Softmax output predictions: $\mathrm{p}(y \mid \boldsymbol{x})$
- GMM density: $\mathrm{q}(\boldsymbol{z}) = \sum_y \mathrm{q}(\boldsymbol{z} \mid y = c)\ \mathrm{q}(y = c)$
- Dataset $(\boldsymbol{X}, Y)$

procedure TRAIN
  train regularized NN $\mathrm{p}(y \mid f_\theta(\boldsymbol{x}))$ with (X, Y)
  for each class c with samples $\boldsymbol{x}_c \subset X$ **do**
    $\mu_c \leftarrow \frac{1}{|\boldsymbol{x}_c|} \sum_{\boldsymbol{x}_c} f_\theta(\boldsymbol{x}_c)$
    $\Sigma_c \leftarrow \frac{1}{|\boldsymbol{x}_c|-1}(f_\theta(\boldsymbol{x}_c) - \mu_c)(f_\theta(\boldsymbol{x}_c) - \mu_c)^T$
    $\pi_c \leftarrow \frac{\sum_{\boldsymbol{x}_c} 1}{|X|}$
  end for
end procedure

function DISENTANGLE_UNCERTAINTY(sample $\boldsymbol{x}$)
  compute feature representation $\boldsymbol{z} = f_\theta(\boldsymbol{x})$
  compute density under GMM: $\mathrm{q}(\boldsymbol{z}) = \sum_y \mathrm{q}(\boldsymbol{z} \mid y)\, \mathrm{q}(y)$ with $\mathrm{q}(\boldsymbol{z} \mid y) \sim \mathcal{N}(\mu_y; \sigma_y), \mathrm{q}(y) = \pi_y$
  compute softmax entropy: $H_p[Y|\boldsymbol{x}]$

  if low density $\mathrm{q}(\boldsymbol{z})$ **then**
    return $(-\mathrm{q}(\boldsymbol{z}), \emptyset)$
  else if high density $\mathrm{q}(\boldsymbol{z})$ **then**
    return $(-\mathrm{q}(\boldsymbol{z}), H_p[Y|\boldsymbol{x}])$
  end if
end function

Disentangling Epistemic & Aleatoric Uncertainty. To quantify epistemic uncertainty, we fit a feature-space density estimator after training. We use GDA, a GMM $\mathrm{q}(y, \boldsymbol{z})$ with a single Gaussian component per class, and fit each class component by computing the empirical mean and covariance, per class, of the feature vectors $z = f_\theta(\boldsymbol{x})$, which are the outputs of the last convolutional layer of the model computed on the training samples $\boldsymbol{x}$. *Note that we do not require OoD data to fit these and unlike Lee et al. [2018b] we use a separate covariance matrix for each class.* Fitting a GDA on the feature space, thus requires no further training and only requires a single forward-pass through the training set.

Evaluation. At test time, we estimate the epistemic uncertainty by evaluating the marginal likelihood of the feature representation under our density $\mathrm{q}(\boldsymbol{z}) = \sum_y \mathrm{q}(\boldsymbol{z} \mid y)\, \mathrm{q}(y)$. To quantify aleatoric uncertainty for in-distribution samples, we use the entropy $\mathrm{H}[Y \mid \boldsymbol{x}, \theta]$ of the softmax distribution $\mathrm{p}(y|\boldsymbol{x}, \theta)$. Note that the softmax distribution thus obtained can be further calibrated using temperature scaling [Guo et al., 2017]. Thus, for a given input, a high feature-space density indicates low epistemic uncertainty (iD), at which point, we can trust the aleatoric estimate from the softmax entropy. The sample can then be either unambiguous (low softmax entropy) or ambiguous (high softmax entropy). Conversely, a low feature density indicates high epistemic uncertainty (OoD), and we cannot trust softmax predictions. The algorithm is depicted in Algorithm 1.

```
# instantiate models
model = create_sensitive_smooth_model()
gda = create_gda()

# train
training_samples, training_labels = load_training_set()
model.fit(training_samples, training_labels)

training_features = model.features(training_samples)
gda.fit(training_features, training_labels)

# test
test_features = model.features(test_sample)

epistemic_uncertainty = -gda.log_density(test_features)

is_ood = epistemic_uncertainty <= ood_threshold
if not is_ood:
  predictions = model.softmax_layer(test_features)
  aleatoric_uncertainty = entropy(predictions)
  return epistemic_uncertainty, aleatoric_uncertainty

# aleatoric uncertainty is only valid for iD
return epistemic_uncertainty, None
```

Listing 3.1: Deep Deterministic Uncertainty Pseudo-Code

3.4.0.1 Computational Complexity

Let N be the number of samples; D, the feature space dimensionality; and C, the number of classes; with $\approx N/C$ samples per class (balanced). For fitting the GMM via GDA: computing the covariance matrix per class requires $\mathcal{O}(C(N/C)D^2) = \mathcal{O}(ND^2)$ complexity. Computing the inverse and determinant of the covariance matrices via the Cholesky decomposition requires $\mathcal{O}(D^3)$ per class. Thus, the total computational cost for GDA is $\mathcal{O}(ND^2 + CD^3)$. Evaluating density of a single point: distance from class means requires $\mathcal{O}(CD)$, and matrix vector multiplications requires $\mathcal{O}(CD^2)$. Hence, the total cost for evaluating density on a single point is $\mathcal{O}(CD^2)$.

3.4.1 Implementation

Here, we describe our reference implementation in more detail. A simple Python pseudocode using a 'scikit-learn'-like API [Buitinck et al., 2013] is shown in Listing 3.1. Note that in order to compute thresholds for low and high density or entropy, we simply use the training set containing iD data. We consider all points having density lower than the 99% quantile as OoD.

Increasing sensitivity. Using residual connections to enforce sensitivity works well in practice when the layer is defined as $\boldsymbol{x}' = \boldsymbol{x} + f(\boldsymbol{x})$. However, there are several places in the network where additional spatial downsampling is done in $f(\cdot)$ (through a strided convolution), and in order to compute the residual operation $\boldsymbol{x}$ needs to be downsampled as well. These downsampling operations are crucial for managing memory consumption and generalization. The way this is traditionally done in ResNets is by introducing an additional function $g(\cdot)$ on the residual branch (obtaining $\boldsymbol{x}' = g(\boldsymbol{x}) + f(\boldsymbol{x})$) which is a strided 1x1 convolution. In practice, the stride is set to 2 pixels, which leads to the output of $g(\cdot)$ only being dependent on the top-left pixel of each 2x2 patch, which reduces sensitivity. We overcome this issue by making an architectural change that improves uncertainty quality without sacrificing accuracy. We use a strided average

pooling operation instead of a 1x1 convolution in $g(\cdot)$. This makes the output of $g(\cdot)$ dependent on all input pixels. Additionally, we use leaky ReLU activation functions, which are equivalent to ReLU activations when the input is larger than 0, but below 0 they compute $p * \boldsymbol{x}$ with $p = 0.01$ in practice. These further improve sensitivity as all negative activations still propagate in the network.

3.5 Empirical Validation

We evaluate DDU on active learning and OoD detection tasks:

Visualizations. We detail toy experiments on how DDU can disentangle epistemic and aleatoric uncertainty and the effect of feature-space regularization in, depicted in Figure 3.2, and on the well-known Two Moons toy dataset in §3.5.1.1.

Active Learning. For active learning [Cohn et al., 1996], we evaluate DDU using MNIST, CIFAR-10 and an ambiguous version of MNIST (Dirty-MNIST).

OoD Detection. Understanding the caveats detailed in §3, we can use OoD detection to evaluate using DDU for estimating epistemic uncertainty and aleatoric uncertainty such that it will be meaningful for active learning as well: we will focus on 'near OoD' datasets such that good OoD detection performance is a good proxy for good epistemic uncertainty performance, and we do not use 'far OoD' datasets as they would not be informative for active learning performance.

Thus, we evaluate DDU's quality of epistemic uncertainty estimation on several OoD detection settings for:

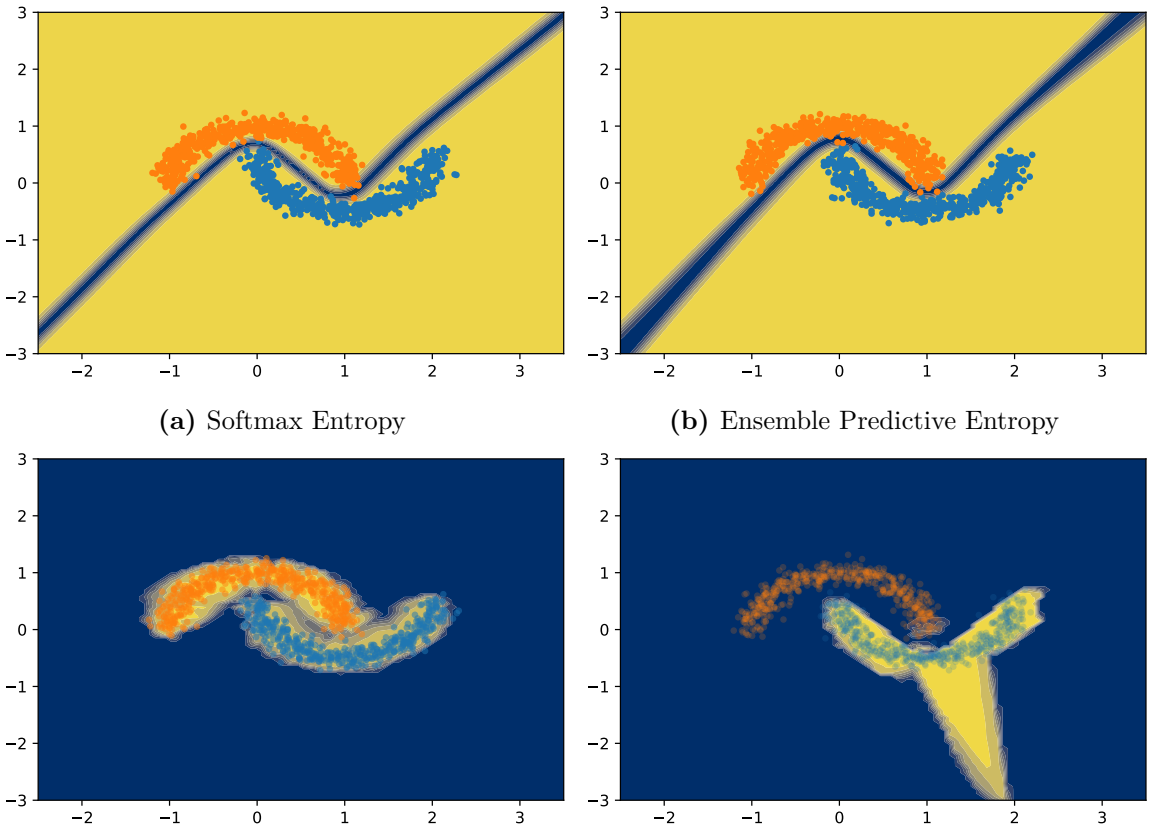
- image classification on CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C, CIFAR-100 vs SVHN/Tiny-ImageNet and ImageNet vs ImageNet-O dataset pairings, where we outperform other deterministic single-forward-pass methods and perform on par with deep ensembles;
- semantic segmentation on Pascal VOC, comparing with a deterministic model, MC Dropout (MCDO) Gal and Ghahramani [2016a] and deep ensembles;
- on the real-world QUBIQ challenge in §C.3.1

Ablations. We ablate feature space density on different model architectures, compare GDA and LDA, examine the effect of the objective mismatch on CIFAR-10, and provide results on additional baselines, with additional ablations in Appendix C.2.

While the focus of this thesis is on data subset selection, the bulk of the experiments for DDU is on OoD detection as these experiments are easier to run and ablate than active learning experiments.

3.5.1 Visualizations

Two toy experiments illustrate the effect of feature-space regularization on the quality of epistemic and aleatoric uncertainty estimation. The first experiment is on a simple 2D toy dataset, and the second experiment is on the MNIST dataset.



(a) Softmax Entropy

(b) Ensemble Predictive Entropy

(c) DDU Feature-space density

(d) FC-Net Feature-space density

Figure 3.15: *Uncertainty on Two Moons dataset.* Blue indicates high uncertainty and yellow indicates low uncertainty. Both the softmax entropy of a single model and the predictive entropy of a deep ensemble are uncertain only along the decision boundary whereas the feature-space density of DDU is uncertain everywhere except on the data distribution (the ideal behavior). However, the feature density of a normal fully connected network (FC-Net) without any inductive biases can't capture uncertainty properly.

3.5.1.1 Two Moons

In this section, we evaluate DDU's performance on a well-known toy setup: the Two Moons dataset. We use scikit-learn's *datasets* package to generate 2000 samples with a noise rate of 0.1. We use a 4-layer fully connected architecture, ResFFN-4-128 with 128 neurons in each layer and a residual connection, following [Liu et al., 2020a]. As an ablation, we also train using a 4-layer fully connected architecture with 128 neurons in each layer, but *without the residual connection*. We name this architecture FC-Net. The input is 2-dimensional and is projected into the 128 dimensional space using a fully connected layer. Using the ResFFN-4-128 architecture we train 3 baselines:

Softmax. We train a single softmax model and use the softmax entropy as the uncertainty metric.

3-Ensemble. We train an ensemble of 3 softmax models and use the predictive entropy of the ensemble as the measure of uncertainty.

DDU. We train a single softmax model applying spectral normalization on the fully connected layers and using the feature density as the measure of model confidence.

Table 3.4: *ECE for Dirty-MNIST test set and AUROC for Dirty-MNIST vs FashionMNIST as proxies for aleatoric and epistemic uncertainty quality respectively.*

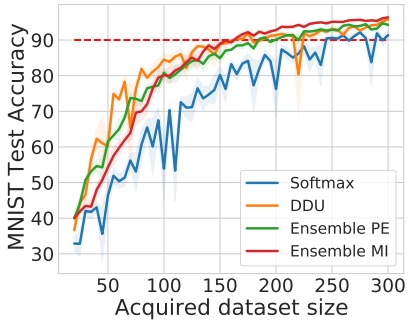
Model	ECE (↓)	AUROC	
		Softmax Entropy (↑)	Feature Density (↑)
LeNet	2.22	84.23	71.41
VGG-16	**2.11**	84.04	89.01
ResNet-18+SN (DDU)	2.34	83.01	**99.91**

Each model is trained using the Adam optimizer for 150 epochs. In Figure 3.15, we show the uncertainty results for all the above 3 baselines. It is clear that both the softmax entropy and the predictive entropy of the ensemble is uncertain only along the decision boundary between the two classes whereas DDU is confident only on the data distribution and is not confident anywhere else. It is worth mentioning that even DUQ and SNGP perform well in this setup and deep ensembles have been known to underperform in the Two-Moons setup primarily due to the simplicity of the dataset causing all the ensemble components to generalize in the same way. Finally, also note that the feature space density of FC-Net without residual connections is not able to capture uncertainty well (Figure 3.15(d)), thereby reaffirming our claim that proper inductive biases are indeed a necessary component to ensure that feature space density captures uncertainty reliably.

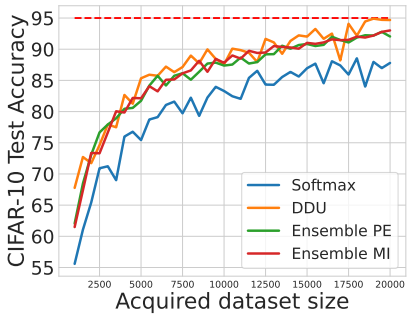
3.5.1.2 Motivational Example in Figure 3.2

As mentioned, in Figure 3.2 we train a LeNet [LeCun et al., 1998], a VGG-16 [Simonyan and Zisserman, 2015] and a ResNet-18 with spectral normalization [He et al., 2016; Miyato et al., 2018] (ResNet-18+SN) on Dirty-MNIST. Figure 3.2(b) shows that the softmax entropy of a deterministic model is unable to distinguish between iD (Dirty-MNIST) and OoD (FashionMNIST [Xiao et al., 2017]) samples as the entropy for the latter heavily overlaps with the entropy for Ambiguous-MNIST samples. However, the feature-space density of the model with our inductive biases in Figure 3.2(c) captures epistemic uncertainty reliably and is able to distinguish iD from OoD samples. The same cannot be said for LeNet or VGG in Figure 3.2(c), whose densities are unable to separate OoD from iD samples. This demonstrates the importance of the inductive bias to ensure the sensitivity and smoothness of the feature space as we further argue below. Finally, Figure 3.2(b) and Figure 3.2(c) demonstrate that our method is able to separate aleatoric from epistemic uncertainty: samples with low feature density have high epistemic uncertainty, whereas those with both high feature density and high softmax entropy have high aleatoric uncertainty—note the high softmax entropy for the most ambiguous Ambiguous-MNIST samples in Figure 3.2(b).

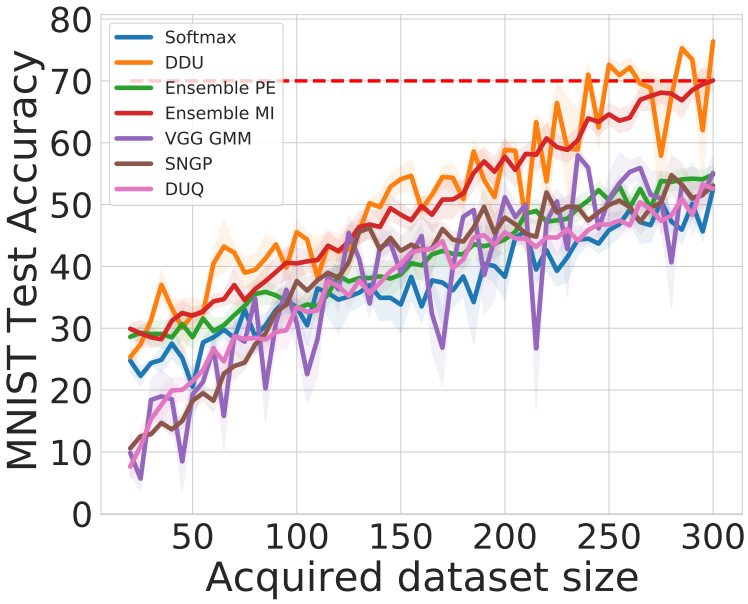
Disentangling Epistemic and Aleatoric Uncertainty Table 3.4 gives a quantitative evaluation of the qualitative results in §3. The AUROC metric reflects the quality of the epistemic uncertainty as it measures the probability that iD and OoD samples can be distinguished, and OoD samples are never seen during training while iD samples are semantically similar to training samples. The ECE metric measures the quality of the aleatoric uncertainty. The softmax outputs capture aleatoric uncertainty well, as expected, and all 3 models obtain similar ECE scores on the Dirty-MNIST test set. However, with an AUROC of around 84% for all the 3 models, on Dirty-



(a) MNIST



(b) CIFAR-10



(c) Dirty-MNIST

Figure 3.16: *Active Learning experiments.* Acquired training set size vs test accuracy. DDU performs on par with deep ensembles.

MNIST vs FashionMNIST, we conclude that softmax entropy is unable to capture epistemic uncertainty well. This is reinforced in Figure 3.2(b), which shows a strong overlap between the softmax entropy of OoD and ambiguous iD samples. At the same time, the feature-space densities of LeNet and VGG-16, with AUROC scores around 71% and 89% respectively, are unable to distinguish OoD from iD samples, indicating that simply using feature-space density without appropriate inductive biases (as seen in [Lee et al., 2018b]) is not sufficient.

Only by fitting a GMM on top of a feature extractor with appropriate inductive biases (DDU) and using its feature density are we able to obtain performance far better (with AUROC of 99.9%*) than the alternatives in the ablation study (see Table 3.4, but this is also noticeable in Figure 3.2(c)).* The entropy of a softmax model can capture aleatoric uncertainty, even without additional inductive biases, but it *cannot* be used to estimate epistemic uncertainty (see §3.2). On the other hand, feature-space density can *only* be used to estimate epistemic uncertainty *when the feature extractor is sensitive and smooth*, as achieved by using a ResNet and spectral normalization in DDU.

3.5.2 Active Learning

We evaluate DDU on three different active learning setups:

1. with clean MNIST samples in the pool set,
2. with clean CIFAR-10 samples in the pool set, and
3. with Dirty-MNIST, having a 1:60 ratio of MNIST to Ambiguous-MNIST samples, in the pool set.

In the first two setups, we compare three baselines as for two moons:

- a ResNet-18 with softmax entropy as the acquisition function,

- DDU trained using a ResNet-18 with feature density as acquisition function, and
- a deep ensemble of 3 ResNet-18s with the predictive entropy (PE) and mutual information (MI) of the ensemble as the acquisition functions.

For Dirty-MNIST, in addition to the above 3 approaches, we also compare to:

- feature density of a VGG-16 instead of ResNet-18+SN as an ablation to see if feature density of a model without inductive biases performs well, as well as
- SNGP [Lee et al., 2020] and DUQ [van Amersfoort et al., 2020] as additional baselines.

For MNIST and Dirty-MNIST, we start with an initial training-set size of 20 randomly chosen MNIST points, and in each iteration, acquire the 5 samples with the highest reported epistemic uncertainty. We re-train the models after each batch acquisition step using Adam [Kingma and Ba, 2015] for 100 epochs and choose the model with the best validation set accuracy. We stop the process when the training set size reaches 300. For CIFAR-10, we start with 1000 samples and go up to 20000 samples with an acquisition size of 500 samples in each step.

MNIST & CIFAR-10. In Figure 3.16(a) and Figure 3.16(b), for regular curated MNIST and CIFAR-10 in the pool set, DDU clearly outperforms the deterministic softmax baseline and is competitive with deep ensembles. For MNIST, the softmax baseline reaches 90% test-set accuracy at a training-set size of 245. DDU reaches 90% accuracy at a training-set size of 160, whereas deep ensemble reaches the same at 185 and 155 training samples with PE and MI as the acquisition functions respectively. Note that DDU is three times faster than a deep ensemble, which needs to train three models independently after every acquisition.

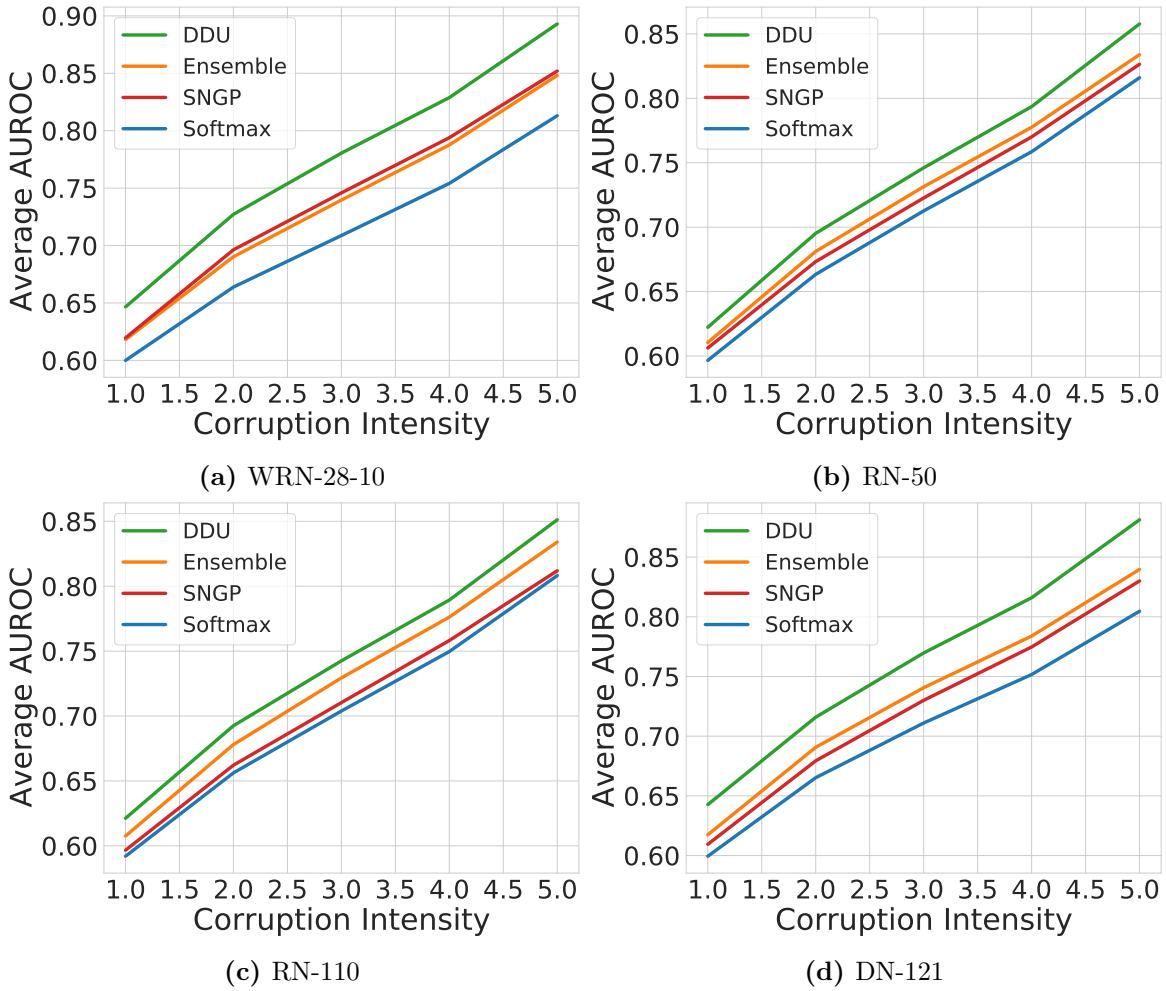
Dirty-MNIST. Real-life datasets often contain observation noise and ambiguous samples. What happens when the pool set contains a lot of such noisy samples having high aleatoric uncertainty? In such cases, it becomes important for models to identify unseen and informative samples with high epistemic uncertainty and not with high aleatoric uncertainty. To study this, we construct a pool set with samples from Dirty-MNIST (see §3.1.1.1). We significantly increase the proportion of ambiguous samples by using a 1:60 split of MNIST to Ambiguous-MNIST (a total of 1K MNIST and 60K Ambiguous-MNIST samples). In Figure 3.16(c), for Dirty-MNIST in the pool set, the difference in the performance of DDU and the deterministic softmax model is stark. While DDU achieves a test set accuracy of 70% at a training set size of 240 samples, the accuracy of the softmax baseline peaks at a mere 50%. In addition, all baselines, including SNGP, DUQ and the feature density of a VGG-16, which fail to solely capture epistemic uncertainty, are significantly outperformed by DDU and the MI baseline of the deep ensemble. However, note that DDU also performs better than deep ensembles with the PE acquisition function. The difference gets larger as the training set size grows: DDU's feature density and deep ensemble's MI solely capture epistemic uncertainty and hence, do not get confounded by iD ambiguous samples with high aleatoric uncertainty.

3.5.3 OoD Detection

Near-OoD detection is an application of epistemic uncertainty quantification: if we do not train on OoD data, we expect near OoD data points to have higher epistemic uncertainty than iD data.

Table 3.5: *OoD detection performance of different baselines using a Wide-ResNet-28-10 architecture with the CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet dataset pairs averaged over 25 runs. SN:* Spectral Normalization, *JP:* Jacobian Penalty. The best deterministic single-forward pass method and the best method overall are in bold for each metric.

Train Dataset	**Method**	**Penalty**	Aleatoric Uncertainty	**Epistemic Uncertainty**	Accuracy (↑)	ECE (↓)	**AUROC**		
							SVHN (↑)	**CIFAR-100** (↑)	**Tiny-ImageNet** (↑)
CIFAR-10	Softmax	-	Softmax Entropy	Softmax Entropy	95.98 ± 0.02	$\mathbf{0.85 \pm 0.02}$	94.44 ± 0.43	89.39 ± 0.06	88.42 ± 0.05
	Energy-based [Liu et al., 2020b]	-		Softmax Density			94.56 ± 0.51	88.89 ± 0.07	88.11 ± 0.06
	DUQ [van Amersfoort et al., 2020]	JP	Kernel Distance	Kernel Distance	94.6 ± 0.16	1.55 ± 0.08	93.71 ± 0.61	85.92 ± 0.35	86.83 ± 0.12
	SNGP [Liu et al., 2020a]	SN	Predictive Entropy	Predictive Entropy	$\mathbf{96.04 \pm 0.09}$	1.8 ± 0.1	94.0 ± 1.3	91.13 ± 0.15	89.97 ± 0.19
	DDU (ours)	**SN**	**Softmax Entropy**	**GDA Density**	95.97 ± 0.03	$\mathbf{0.85 \pm 0.04}$	$\mathbf{97.86 \pm 0.19}$	$\mathbf{91.34 \pm 0.04}$	$\mathbf{91.07 \pm 0.05}$
	5-Ensemble [Lakshminarayanan et al., 2017]	-	Predictive Entropy	Predictive Entropy	$\mathbf{96.59 \pm 0.02}$	$\mathbf{0.76 \pm 0.03}$	97.73 ± 0.31	$\mathbf{92.13 \pm 0.02}$	90.06 ± 0.03
				Mutual Information			97.18 ± 0.19	91.33 ± 0.03	90.90 ± 0.03
					Accuracy (↑)	ECE (↓)	**SVHN** (↑)		**Tiny-ImageNet** (↑)
CIFAR-100	Softmax	-	Softmax Entropy	Softmax Entropy	80.26 ± 0.06	4.62 ± 0.06	77.42 ± 0.57		81.53 ± 0.05
	Energy-based [Liu et al., 2020b]	-		Softmax Density			78 ± 0.63		81.33 ± 0.06
	SNGP [Liu et al., 2020a]	SN	Predictive Entropy	Predictive Entropy	80.00 ± 0.11	4.33 ± 0.01	85.71 ± 0.81		78.85 ± 0.43
	DDU (ours)	**SN**	**Softmax Entropy**	**GMM Density**	$\mathbf{80.98 \pm 0.06}$	$\mathbf{4.10 \pm 0.08}$	$\mathbf{87.53 \pm 0.62}$		$\mathbf{83.13 \pm 0.06}$
	5-Ensemble [Lakshminarayanan et al., 2017]	-	Predictive Entropy	Predictive Entropy	$\mathbf{82.79 \pm 0.10}$	$\mathbf{3.32 \pm 0.09}$	79.54 ± 0.91		82.95 ± 0.09
				Mutual Information			77.00 ± 1.54		82.82 ± 0.04



(a) WRN-28-10 **(b)** RN-50

(c) RN-110 **(d)** DN-121

Figure 3.17: AUROC vs corruption intensity averaged over all corruption types in CIFAR-10-C for 4 architectures. More details in §3.5.3 and more ablations in §C.1 in the appendix.

3.5.3.1 Image Classification

We evaluate CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C, CIFAR-100 vs SVHN/Tiny-ImageNet and ImageNet vs ImageNet-O as iD vs OoD dataset pairs for this experiment [Krizhevsky, 2009; Netzer et al., 2011; Deng et al., 2009; Hendrycks and Dietterich, 2019]. We also evaluate DDU on different architectures: Wide-ResNet-28-10, Wide-ResNet-50-2, ResNet-50, ResNet-110 and DenseNet-121 [Zagoruyko and Komodakis, 2016; He et al., 2016; Huang et al., 2017]. The training setup is described in §C.1.2. In addition to using softmax entropy of a deterministic model (*Softmax*) for both aleatoric and epistemic uncertainty, we also compare with the following **baselines** that do not require training or fine-tuning on OoD data:

Energy-based model [Liu et al., 2020b]: We use the softmax entropy of a deterministic model as aleatoric uncertainty and the unnormalized softmax density (the logsumexp of the logits) as epistemic uncertainty *without* regularization to avoid feature collapse. We only compare with the version that does not train on OoD data.

DUQ [van Amersfoort et al., 2020] & SNGP [Liu et al., 2020a]: We compare with the state-of-the-art deterministic methods for uncertainty quantification including DUQ and SNGP. For SNGP, we use the exact predictive covariance computation.

Table 3.6: *Pascal VOC validation set mIoU and runtime in milliseconds averaged over 10 forward passes.* For MC Dropout, we perform 5 stochastic forward passes.

Baseline	Softmax	MC Dropout	Deep Ensemble	**DDU**
mIoU	78.53	78.61	78.47	78.53
Runtime (ms)	275.48 ± 1.91	1576.75 ± 1.56	875.87 ± 0.79	$\mathbf{263.83 \pm 2.79}$

We measure uncertainty via the entropy of the average of the MC softmax samples. For DUQ, we use the closest kernel distance. Note that for CIFAR-100, DUQ's one-vs-all objective did not converge during training and hence, we do not include the DUQ baseline for CIFAR-100.

5-Ensemble: We use an ensemble of 5 networks with the same architecture and compute the predictive entropy of the ensemble as both epistemic and aleatoric uncertainty and mutual information as epistemic uncertainty.

Results. Table 3.5 presents the AUROC for Wide-ResNet-28-10 models on CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet along with their respective test set accuracy and ECE post temp-scaling (additional calibration scores in §C.3.2 and comparison with more baselines in §3.5.4). The equivalent results for other architectures: ResNet-50/110 and DenseNet-121 can be found in Table C.1, Table C.2 and Table C.3 in the appendix. Note that for DDU, post-hoc calibration with temperature scaling [Guo et al., 2017], is simple as it does not affect the GMM density. We also plot the AUROC averaged over corruption types vs corruption intensity for CIFAR-10 vs CIFAR-10-C in Figure 3.17, with AUROC plots per corruption type in Figure C.2, Figure C.3, Figure C.4 and Figure C.5 of the appendix. Finally, in Table 3.7, we present AUROC for models trained on ImageNet.

For OoD detection, *DDU outperforms all other deterministic single-forward-pass methods, DUQ, SNGP and the energy-based model approach from [Liu et al., 2020b], on CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet, CIFAR-10 vs CIFAR-10-C and CIFAR-100 vs SVHN/Tiny-ImageNet, often performs on par with state-of-the-art deep ensembles—and even performing better in a few cases.* This holds true for all the architectures we experimented on. Similar observations can be made on ImageNet vs ImageNet-O as well. Importantly, the great performance in OoD detection comes without compromising on the single-model test set accuracy in comparison to other deterministic methods.

3.5.3.2 Semantic Segmentation

In this section, we apply DDU to the task of semantic segmentation on Pascal VOC 2012 Everingham et al. [2010], comparing with a vanilla softmax model, MC Dropout and deep ensembles. Semantic segmentation [Long et al., 2015], classifies every pixel of a given to one of a fixed set of classes. Since different classes can have different levels of representation in a segmentation dataset, it forms a classic example of a problem with class imbalance, thereby requiring reliable estimates of epistemic uncertainty. Furthermore, due to the computationally heavy nature of semantic segmentation, classic uncertainty quantification approaches like MC Dropout and deep ensembles are often impractical in real-world applications.

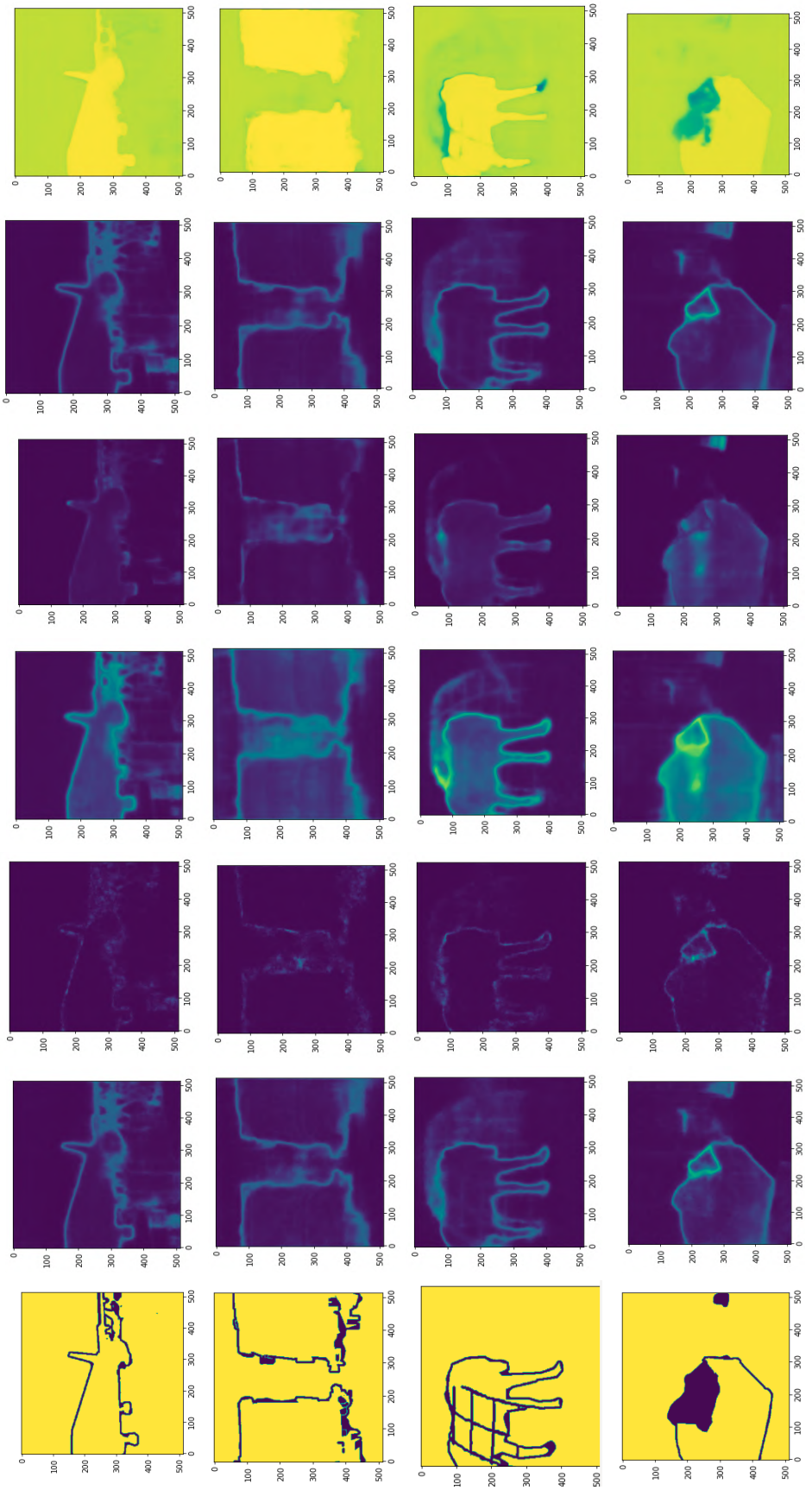


Figure 3.18: *Visualization of uncertainty baselines on four PASCAL VOC validation samples (**rows**).* **Columns**: **(a)** shows pixel-wise accuracy; **(b), (c)** predictive entropy (PE) and mutual information (MI) obtained for MC Dropout (MCD); **(d), (e)** for deep ensembles; **(f)** per-pixel softmax entropy, *the aleatoric estimate of DDU*; and **(g)** feature density, *the epistemic component of DDU*. For all but (g): the brighter, the more uncertain, whereas DDU's density (g) captures certainty: hence, the brighter, the more certain.

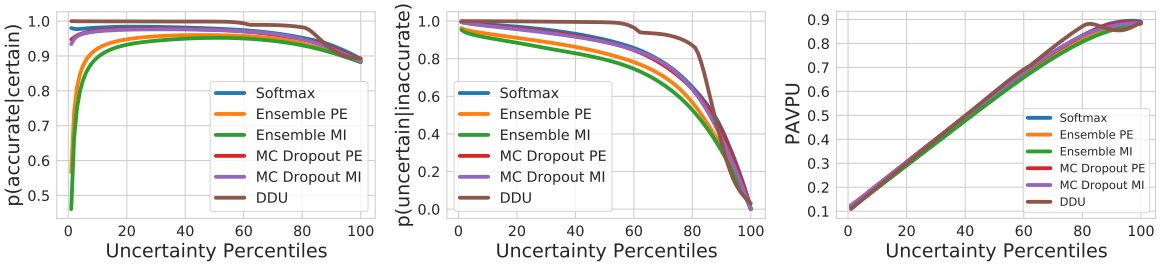


Figure 3.19: p(accurate | certain), p(uncertain | inaccurate) *and* PAvPU *evaluated on PASCAL VOC validation set.* DDU outperforms all other baselines.

Pixel-Independent Class-Wise Means and Covariances. As each pixel has a corresponding prediction in semantic segmentation, it is natural to ask if the Gaussian means and covariance matrices need to be computed per pixel. To examine this, in Figure C.1 of §C.1.3, we plot the L2 distances between feature space means of all pairs of classes obtained from a DeepLab-v3+ Chen et al. [2017] model with a ResNet-101 backbone for two "distant" pixels. We observe that pixels of the same class are much closer in the feature space than pixels of different classes, irrespective of their location in the image. In spirit of a new simple baseline, we thus compute the Gaussian means and covariances per class, taking each pixel as a separate data point.

Architecture, Training and Evaluation Metrics. As mentioned above, we evaluate DDU on Pascal VOC 2012 and compare to a vanilla softmax model, MC Dropout with 5 forward passes at test time, and a deep ensembles with 3 members. We use DeepLab-v3+ with a ResNet-101 backbone as the model architecture. Additional training details are in §C.1.3. Finally, to evaluate the uncertainty estimates, we use patch-based metrics proposed in Mukhoti and Gal [2018]: p(accurate | certain), p(uncertainty | inaccurate) and the *Patch Accuracy vs Patch Uncertainty* PAvPU. p(accurate | certain) computes the probability of the model being accurate given that it is confident. Similarly, p(uncertainty | inaccurate) measures probability of the model being uncertain given that it is inaccurate, and $PAvPU$ computes the probability of the model being confident on accurate predictions and uncertain on inaccurate ones, so the accuracy depending on the uncertainty threshold similar to a rejection plot. Ideally, high values for these metrics indicate better uncertainty estimates in segmentation. Furthermore, note that these metrics can be computed at different thresholds of uncertainty (defining if a model is certain or not).

Results and Discussion. In Figure 3.19, we present the above 3 metrics for all segmentation baselines evaluated on the Pascal VOC validation set. We also report the val set accuracy and runtime of a single forward pass in Table 3.6. Finally, we visualize uncertainty estimates from each baseline in Figure 3.18. Firstly, from Table 3.6, it is clear that DDU has the runtime of a deterministic model which is significantly faster than both MC Dropout and deep ensembles. Also note that DDU's mIoU is the same as that of the vanilla softmax model. Secondly, from Figure 3.19, we see that DDU consistently performs better on all 3 evaluation metrics compared to the other baselines. Finally, Figure 3.18 qualitatively validates that DDU's feature-space density captures epistemic uncertainty while the softmax entropy captures aleatoric

uncertainty. For DDU, for the first two samples (first two rows, Figure 3.18(g)), the epistemic uncertainty is not high and only aleatoric uncertainty is captured along edges of objects. However, for the last sample (4^{th} row, Figure 3.18(g)), the epistemic uncertainty is high for a relatively large patch on the image which is inaccurately predicted by the model as well. Note that only DDU's feature density is significantly lower for that entire region, whereas softmax entropy does not capture high uncertainty there and is only high along the edges. These observations are in line with [Kendall and Gal, 2017]: aleatoric uncertainty is high on edges of objects as they correspond to regions of high ambiguity and noise; on the other hand, epistemic uncertainty is high for regions of the image which are previously unseen.

3.5.4 Ablations

Additional ablations for the CIFAR-10/100 experiments are detailed in §C.2, Table C.4 and C.5. We highlight a few results here:

Feature Density. These tables along with observations in Table 3.7, show that *the feature density of a VGG-16 (i.e. without residual connections and spectral normalization) is unable to beat a VGG-16 ensemble, whereas a Wide-ResNet-28-10 with spectral normalization outperforms its corresponding ensemble in almost all the cases.* This result further validates the importance of having a regularized feature space on the model to obtain smoothness and sensitivity. Also note that, even without spectral normalization, a Wide-ResNet-28 has residual connections built into its model architecture, which can be a contributing factor towards good performance as residual connections make the model sensitive to changes in the input space.

GDA vs. LDA. We also provide an ablation using LDA [Lee et al., 2018b], which uses a shared covariance matrix over all classes, instead of GDA with covariance matrices per class. The resulting AUROC for Wide-ResNet-28-10 trained on CIFAR-10/100 and for Wide-ResNet-50-2 and ResNet-50 trained on ImageNet in Table 3.8 in §C.2. LDA only outperforms GDA when using SVHN as an OoD dataset. In all other cases, GDA obtains significantly higher AUROC, thereby indicating the advantage of modeling density using individual covariance matrices per class.

Objective Mismatch with Wide-ResNet-28-10 on CIFAR-10. We further validate Proposition 3.3 by running an ablation on Wide-ResNet-28-10 on CIFAR-10. Table 3.9 shows that the feature-space density estimator indeed performs worse than the softmax layer for aleatoric uncertainty (accuracy and ECE).

Additional Baselines. We provide an ablation with additional baselines on OoD detection for comparison with DDU. In particular, we provide comparisons with Feature Space Singularity (FSSD) [Huang et al., 2021], Batch Ensemble (BE) [Wen et al., 2020] and SWAG [Maddox et al., 2019] using Wide-ResNet-28-10 as additional recent baselines. Huang et al. [2021] computes the distance to the centroid of noise samples in feature-space together with input perturbations, like in [Lee et al., 2018b]. Noise samples count as 'far OoD'. [Wen et al., 2020] and [Maddox et al., 2019] are computationally cheaper ensembling methods.

We also use the Wide-ResNet-28-10 feature extractor trained using SNGP loss and fit DDU (i.e., GDA) on its feature space. Since SNGP also uses a sensitive smooth feature space with residual connections and spectral normalization, its feature space makes for a good candidate to apply DDU. In Table 3.10, we provide the AUROC

scores for models trained on CIFAR-10 and CIFAR-100. Broadly, DDU outperforms all competitive baselines. Additionally, we observe a broad improvement in AUROC when DDU is applied on the SNGP feature extractor as compared to vanilla SNGP. However, DDU on a feature extractor trained using softmax loss is still superior to DDU on the SNGP feature extractor.

3.6 Comparison to Prior Work

Several existing approaches model uncertainty using feature-space density but require fine-tuning on OoD data. This chapter has identified feature collapse and objective mismatch as possible reasons for this.

Among these approaches, we have already discussed Mahalanobis distances, DUQ, and SNGP [Lee et al., 2018b; van Amersfoort et al., 2020; Lee et al., 2020] and the important findings they provide. In this section, we contrast them to the approach presented in this chapter. For one, the best results of Lee et al. [2018b] require input perturbations, ensembling GMM densities from multiple layers, and fine-tuning on OoD hold-out data. Lee et al. [2018b] do not discuss any constraints which the ResNet feature encoder should satisfy, and therefore, are vulnerable to feature collapse—we recall that in Figure 3.2(c), for example, the feature density of a LeNet and a VGG are unable to distinguish OoD from iD samples. Our method also improves upon van Amersfoort et al. [2020] and Liu et al. [2020a] by alleviating the need for additional hyperparameters: DDU only needs minimal changes from the standard softmax setup to outperform DUQ and SNGP on uncertainty benchmarks, and our GMM parameters are optimized for the already trained model using the training set. The insights in §3.2 might also explain why Liu et al. [2020a] found that an ablation that uses *the softmax entropy instead of the feature-space density* of a deterministic network with bi-Lipschitz constraints underperforms.

Among other related works—there are many, and we can only highlight very few here—Postels et al. [2020], Liu et al. [2020b], and [Winkens et al., 2020] are the most relevant: [Postels et al., 2020] propose a density-based estimation of aleatoric and epistemic uncertainty. Similar to [Lee et al., 2018b], they do not constrain their pre-trained ResNet encoder. They do discuss feature collapse though, noting that they do not address this problem. They also do not consider the objective mismatch that arises (see Proposition 3.3 below) and use a single estimator for both epistemic and aleatoric uncertainty. Consequently, they report worse epistemic uncertainty: 74% AUROC on CIFAR-10 vs SVHN, which we show to considerably fall behind modern approaches for uncertainty estimation in deep learning in §3.5. Indeed, Postels et al. [2020] report that in their case deeper layers provide better aleatoric uncertainty while shallower layers provide better epistemic uncertainty. This might be a result of the objective mismatch and not regularizing the feature space using appropriate inductive biases Likewise, Liu et al. [2020b] compute an unnormalized density based on the softmax logits without taking into account the need for inductive biases to ensure smoothness and sensitivity of the feature space. Finally, Winkens et al. [2020] use contrastive training on the feature extractor before estimating the feature-space density. Our method is orthogonal to this work as we restrict ourselves to the supervised setting and show that the inductive biases that encourage bi-Lipschitzness [van Amersfoort et al., 2020; Liu et al., 2020a] are sufficient for the feature-space density to more reliably capture epistemic uncertainty.

Overall, compared to other methods that require held-out "OoD data" for outlier exposure, DDU does not require training or fine-tuning with OoD data in any form.

3.7 Discussion

We began this chapter by looking at uncertainty quantification from a conceptual view, comparing how active learning and OoD detection use uncertainty quantification. Having identified potential pitfalls when using the predictive entropy of deep ensembles or the softmax entropy of deterministic models as proxy for epistemic uncertainty—these potential pitfalls likely apply in general when using the predictive distribution to measure epistemic uncertainty—we investigated using feature-space density as a proxy for epistemic uncertainty and the entropy of the predictive distribution for aleatoric uncertainty, and looked at a possible objective mismatch in detail.

Based on these insights and detailed experiments, a simple method, DDU, was proposed that can obtain reliable epistemic and aleatoric uncertainty estimates for single-pass, deterministic models. By fitting a GDA to estimate feature-space density after training an off-the-shelf neural network with appropriate inductive biases: residual connections and spectral normalization [Lee et al., 2018b; Liu et al., 2020a], our method was able to outperform state-of-the-art deterministic single-pass uncertainty methods in active learning and OoD detection, while performing as well as deep ensembles in several settings. Hence, DDU provides a very simple method that presents an alternative to deep ensembles without requiring the complexities or computational cost of deep ensembles while still providing reliable uncertainty quantification.

This is crucial in applications like active learning, which require a reliable estimate of epistemic uncertainty, but reliable uncertainty quantification is also an important requirement to make deep neural nets safe for deployment. Thus, we hope the insights from this chapter can help increase safety, reliability and trust in AI in the future.

Table 3.7: *OoD detection performance of different baselines using ResNet-50, Wide-ResNet-50-2 and VGG-16 architectures on ImageNet vs ImageNet-O [Hendrycks et al., 2021].* Best AUROC scores are marked in bold.

Model	Accuracy (↑)		ECE (↓)		**AUROC (↑)**				
	Deterministic	3-Ensemble	Deterministic	3-Ensemble	**Softmax Entropy**	**Energy-based Model**	**DDU**	**3-Ensemble PE**	**3-Ensemble MI**
ResNet-50	74.8 ± 0.05	76.01	2.08 ± 0.11	2.07	51.42 ± 0.61	55.76 ± 0.81	$\mathbf{71.29 \pm 0.08}$	60.3	62.43
Wide-ResNet-50-2	76.75 ± 0.11	77.58	1.18 ± 0.07	1.22	52.71 ± 0.23	57.13 ± 0.4	$\mathbf{73.12 \pm 0.19}$	60.45	64.81
VGG-16	72.48 ± 0.02	73.54	2.62 ± 0.11	2.59	50.67 ± 0.22	52.04 ± 0.23	54.32 ± 0.14	58.74	**60.56**

Table 3.8: *LDA vs GDA ablation for OoD detection performance using Wide-ResNet-50-2, ResNet-50, Wide-ResNet-50-2 architectures (depending on dataset) on CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet, CIFAR-100 vs SVHN/Tiny-ImageNet, and ImageNet vs ImageNet-O [Hendrycks et al., 2021].* Best AUROC (↑) scores are marked in bold. GDA performs better, except with SVHN as OoD dataset.

Model	WRN-28-10			WRN-28-10		WRN-50-2	RN-50
iD	CIFAR-10			CIFAR-100		ImageNet	
OoD	SVHN	CIFAR-100	Tiny-ImageNet	SVHN	Tiny-ImageNet	ImageNet-O	
LDA (Maha Lee et al. [2018b])	**98.41 ± 0.09**	82.90 ± 0.23	82.48 ± 0.25	**92.53 ± 0.62**	68.86 ± 0.13	64.19 ± 0.23	61.68 ± 0.14
GDA (DDU, ours)	97.86 ± 0.19	**91.34 ± 0.04**	**91.07 ± 0.05**	87.53 ± 0.62	**83.13 ± 0.06**	**73.12 ± 0.19**	**71.29 ± 0.08**

Table 3.9: *Objective Mismatch Ablation with WideResNet-28-10 models with and without spectral normalization on CIFAR-10.* While the GDA objective performs much better than cross-entropy objective for feature-space density/epistemic uncertainty estimation, it underperforms for aleatoric uncertainty estimation: both accuracy and in particular ECE are much worse than a regular softmax layer. Averaged over 25 runs.

Model	Prediction Source	Accuracy in % (↑)	ECE (↓)
WideResNet-28-10	Softmax	95.98 ± 0.02	2.29 ± 0.02
	GMM	95.86 ± 0.02	4.13 ± 0.02
WideResNet-28-10+SN	Softmax	95.97 ± 0.03	2.23 ± 0.03
	GMM	95.88 ± 0.02	4.12 ± 0.02

Table 3.10: OoD detection ablation with WRN-28-10 model with additional baselines, FSSD [Huang et al., 2021], Batch Ensemble (BE) [Wen et al., 2020] and SWAG [Maddox et al., 2019] as well as using DDU with a feature extractor trained on SNGP. For comparison, we also provide performance for vanilla SNGP, deep ensemble and DDU.

Train Dataset	Method	AUROC (↑)		
		SVHN	***CIFAR-100***	***Tiny-ImageNet***
CIFAR-10	FSSD [Huang et al., 2021]	97.24	89.88	90.23
	BE [Wen et al., 2020]	95.36	87.63	88.14
	SWAG [Maddox et al., 2019]	96.37	90.33	90.24
	SNGP [Liu et al., 2020a]	94.0 ± 1.3	91.13 ± 0.15	89.97 ± 0.19
	5-Ensemble [Lakshminarayanan et al., 2017]	97.73 ± 0.31	$\mathbf{92.13 \pm 0.02}$	90.06 ± 0.03
	SNGP + DDU	96.47 ± 0.7	89.97 ± 0.13	90.3 ± 0.12
	DDU (Ours)	$\mathbf{97.86 \pm 0.19}$	91.34 ± 0.04	$\mathbf{91.07 \pm 0.05}$
		SVHN		***Tiny-ImageNet***
CIFAR-100	FSSD [Huang et al., 2021]	**87.64**		82.2
	BE Wen et al. [2020]	86.44		78.33
	SWAG Maddox et al. [2019]	81.41		81.67
	SNGP Liu et al. [2020a]	85.71 ± 0.81		78.85 ± 0.43
	5-Ensemble Lakshminarayanan et al. [2017]	79.54 ± 0.91		82.95 ± 0.09
	SNGP + DDU	87.34 ± 0.76		79.62 ± 0.36
	DDU (Ours)	87.53 ± 0.62		$\mathbf{83.13 \pm 0.06}$

Batch learning refined,
With correlations in mind,
Knowledge intertwined.

4

Diverse Batch Acquisition for Bayesian Active Learning

In practical active learning applications, instead of single data points, batches of data points are acquired during each acquisition step to reduce the number of times the model is retrained and expert-time is requested. Reasons for this are that model retraining becomes a computational bottleneck for larger models and expert time is expensive. Consider, for example, the effort that goes into commissioning a medical specialist to label a single MRI scan, then waiting until the model is retrained, and then commissioning a new medical specialist to label the next MRI scan, and the extra amount of time this takes.

In Gal et al. [2017], *batch acquisition*, i.e. the acquisition of multiple points, takes the top `K` points with the highest BALD acquisition score. This naive approach leads to acquiring points that are individually very informative, but not necessarily so jointly. See Figure 4.1 for such a batch acquisition of BALD in which it performs poorly whereas scoring points jointly ('BatchBALD') can find *batches* of informative data points. Similarly, Figure 4.3 provides additional anecdotal evidence that the naive approach to batch acquisition can lead to poor acquisitions. Figure 4.2 shows how a dataset consisting of repeated MNIST digits (with added Gaussian noise) leads BALD to perform worse than random acquisition while BatchBALD sustains good performance.

Naively finding the best batch to acquire requires enumerating all possible subsets within the available data, which is intractable as the number of potential subsets grows exponentially with the acquisition size `K` and the size of available points to choose from. Instead, we develop a greedy algorithm that selects a batch in linear time, and show that it is at worst a $1 - 1/e$ approximation to the optimal choice for our acquisition function.

The main contributions of this chapter are:

1. *BatchBALD*, a data-efficient active learning method that acquires *sets* of high-dimensional image data, leading to improved data efficiency and reduced total run time, section 4.1;
2. a greedy algorithm to select a batch of points efficiently, section 4.1.1; and
3. an estimator for the acquisition function that scales to larger acquisition sizes and to datasets with many classes, section 4.1.2.

We provide two open-source implementations of BatchBALD in https://github.com/BlackHC/BatchBALD and https://github.com/BlackHC/BatchBALD_redux.

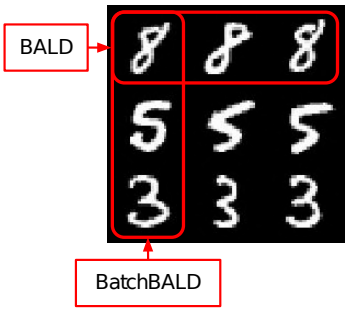


Figure 4.1: *Idealized acquisitions of BALD and BatchBALD.* If a dataset were to contain many (near) replicas for each data point, then BALD would select all replicas of a single informative data point at the expense of other informative data points, wasting data efficiency.

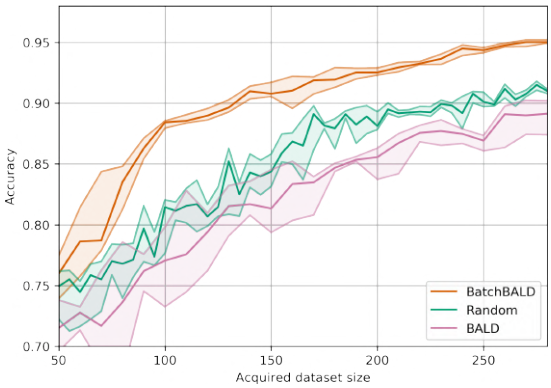


Figure 4.2: *Performance on* Repeated MNIST *with acquisition size 10.* See section 4.2.1 for further details. BatchBALD outperforms BALD while BALD performs worse than random acquisition due to the replications in the dataset.

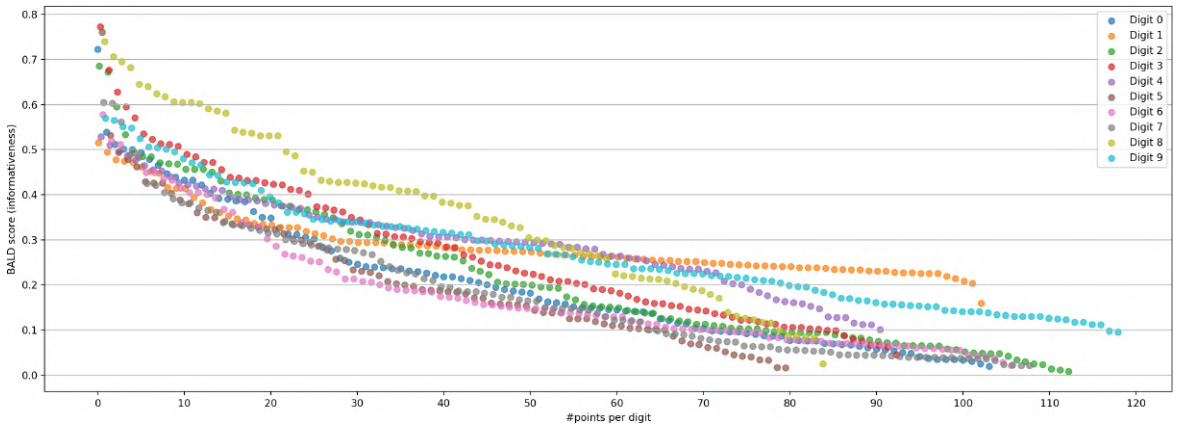


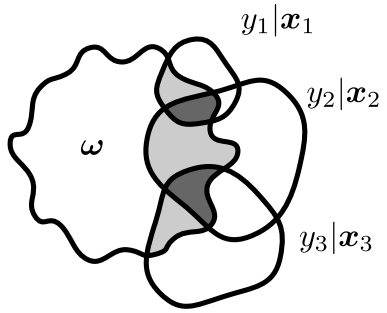
Figure 4.3: *BALD scores for 1000 randomly-chosen points from the MNIST dataset (handwritten digits).* The points are color-coded by digit label and sorted by score. The model used for scoring has been trained to 90% accuracy first. If we were to pick the top scoring points (e.g. scores above 0.6), most of them would be 8s, even though we can assume that after acquiring the first couple of them our model would consider them less informative than other available data. Points are slightly jittered on the x-axis by digit label to avoid overlaps.

4.1 BatchBALD

We propose *BatchBALD* as an extension of BALD whereby we jointly score points by estimating the mutual information between a *joint of multiple data points* and the model parameters:[1]

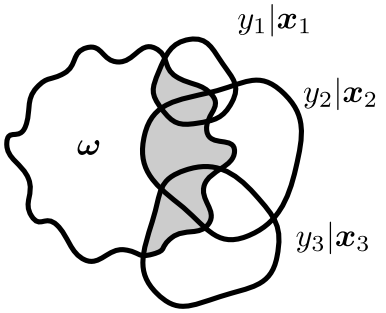
$$a_{\texttt{BatchBALD}}\left(\boldsymbol{x}_{1..K}^{\text{acq}}, \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})\right) = \mathrm{I}[Y_{1..K}^{\text{acq}}; \boldsymbol{\Omega} \mid \boldsymbol{x}_{1..K}^{\text{acq}}, \mathcal{D}^{\text{train}}]. \tag{4.1}$$

[1] We use the notation $\mathrm{I}[\boldsymbol{x}, y; \boldsymbol{z} \mid c]$ to denote the mutual information between the *joint of the random variables* $\boldsymbol{x}, y$ and the random variable $\boldsymbol{z}$ conditioned on c.



$$\sum_i \mathrm{I}[Y_i; \boldsymbol{\Omega} \mid \boldsymbol{x}_i, \mathcal{D}^{\text{train}}] = \sum_i \mu^*[Y_i \cap \boldsymbol{\Omega}]$$

(a) BALD



$$\mathrm{I}[Y_{1..n}; \boldsymbol{\Omega} \mid \boldsymbol{x}_{1..n}, \mathcal{D}^{\text{train}}] = \mu^*\left[\bigcup_i Y_i \cap \boldsymbol{\Omega}\right]$$

(b) BatchBALD

Figure 4.4: *Intuition behind* BALD *and* BatchBALD *using I-diagrams [Yeung, 1991]. BALD* overestimates the joint mutual information. *BatchBALD*, however, takes the overlap between variables into account and will strive to acquire a better cover of $\boldsymbol{\Omega}$. Areas contributing to the respective score are shown in gray, and areas that are double-counted in dark gray.

This builds on the insight that independent selection of a batch of data points leads to data inefficiency as correlations between data points in an acquisition batch are not taken into account.

To understand how to compute the mutual information between a set of points and the model parameters, we express $\boldsymbol{X}^{\text{acq}}_{1..\mathtt{K}}$, and $Y^{\text{acq}}_{1..\mathtt{K}}$ through joint random variables in a product probability space and use the definition of the mutual information for two random variables:

$$\mathrm{I}[Y^{\text{acq}}_{1..\mathtt{K}}; \boldsymbol{\Omega} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}] = \mathrm{H}[Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}]. \tag{4.2}$$

Intuitively, the mutual information between two random variables can be seen as the intersection of their information content. In fact, Yeung [1991] shows that a signed measure μ^* can be defined for discrete random variables X, Y, such that $\mathrm{I}[X;Y] = \mu^*[X \cap Y]$, $\mathrm{H}[X,Y] = \mu^*[X \cup Y]$, $\mathrm{H}[X \mid Y] = \mu^*[X \setminus Y]$, and so on, where we identify random variables with their counterparts in information space.

Using this perspective, BALD can be viewed as the sum of individual intersections $\sum_i \mu^*[Y_i \cap \boldsymbol{\Omega}]$, which double counts overlaps between the Y_i. Naively extending BALD to the mutual information between $Y^{\text{acq}}_{1..\mathtt{K}} | \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}$ and $\boldsymbol{\Omega}$, which is equivalent to $\mu^*[\bigcap_i Y_i \cap \boldsymbol{\Omega}]$, can lead to selecting *similar* data points instead of diverse ones under maximization. BatchBALD, on the other hand, takes overlaps into account by computing $\mu^*[\bigcup_i Y_i \cap \boldsymbol{\Omega}]$ and is more likely to acquire a more diverse cover under maximization:

$$\mathrm{I}[Y^{\text{acq}}_{1..\mathtt{K}}; \boldsymbol{\Omega} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}] = \mathrm{H}[Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}] - \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})} \mathrm{H}[Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \boldsymbol{\omega}, \mathcal{D}^{\text{train}}] \tag{4.3}$$

$$= \mu^*\left[\bigcup_i y_i\right] - \mu^*\left[\bigcup_i Y_i \setminus \boldsymbol{\Omega}\right] = \mu^*\left[\bigcup_i Y_i \cap \boldsymbol{\Omega}\right] \tag{4.4}$$

This is depicted in Figure 4.4 and also motivates that $a_{\texttt{BatchBALD}} \leq a_{\texttt{BALD}}$, which we prove in Appendix D.2. For acquisition size 1, BatchBALD and BALD are equivalent.

Algorithm 2 Greedy BatchBALD $1 - 1/e$-approximate algorithm

Input: acquisition size b, unlabeled dataset $\mathcal{D}_{\text{pool}}$, model parameters $p(\boldsymbol{\omega} \mid \mathcal{D}_{\text{train}})$

$A_0 \leftarrow \emptyset$

for $n \leftarrow 1$ **to** b **do**

  foreach $\boldsymbol{x} \in \mathcal{D}_{pool} \setminus A_{n-1}$ **do** $s_{\boldsymbol{x}} \leftarrow a_{\texttt{BatchBALD}}\left(A_{n-1} \cup \{\boldsymbol{x}\}, p(\boldsymbol{\omega} \mid \mathcal{D}_{\text{train}})\right)$

  $\boldsymbol{x}_n \leftarrow \underset{\boldsymbol{x} \in \mathcal{D}_{\text{pool}} \setminus A_{n-1}}{\arg\max}\ s_{\boldsymbol{x}}$

  $A_n \leftarrow A_{n-1} \cup \{\boldsymbol{x}_n\}$

end

Output: acquisition batch $A_n = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}$

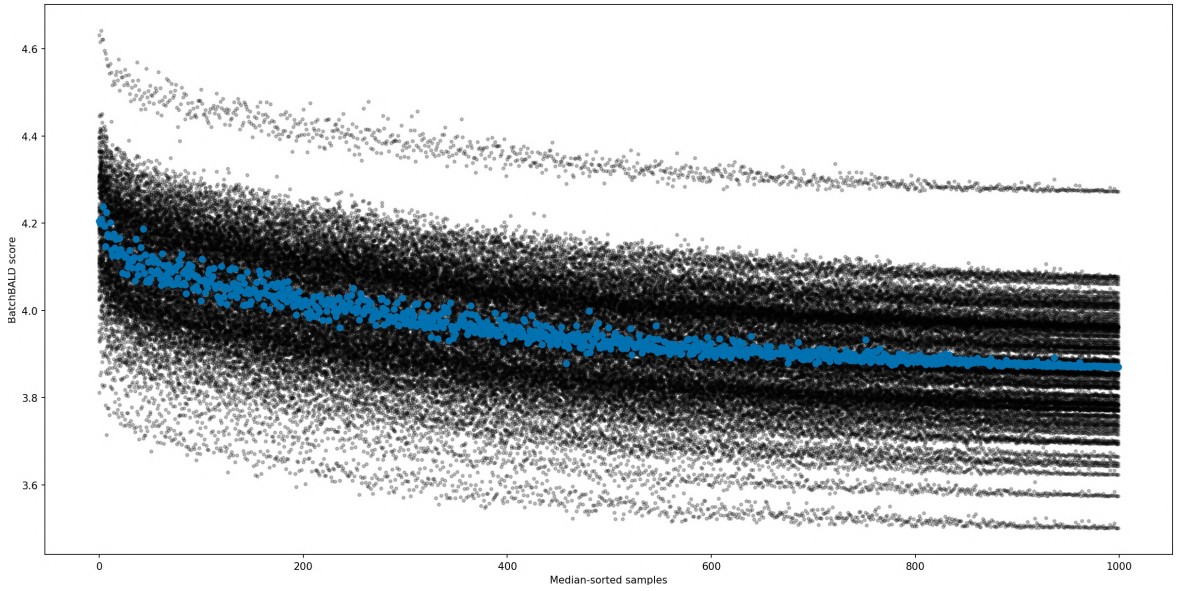


Figure 4.5: *Why **consistent MC dropout** is necessary: BatchBALD scores for different sets of 100 sampled model parameters.* This shows the BatchBALD scores for 1000 randomly selected points from the pool set while selecting the 10th point in a batch for an MNIST model that has already reached 90% accuracy. The scores for a single set of 100 model parameters (randomly chosen) are shown in blue. The BatchBALD estimates show strong banding with the score differences between different sets of sampled parameters being larger than the differences between different data points within a specific set of model parameters. Without consistent sampling, the arg max would essentially be randomly sampled and not be informative.

4.1.1 Greedy Approximation Algorithm for BatchBALD

To avoid the combinatorial explosion that arises from jointly scoring subsets of points, we introduce a greedy approximation for computing BatchBALD, depicted in Algorithm 2. In Appendix D.1, we prove that $a_{\texttt{BatchBALD}}$ is submodular, which means the greedy algorithm is $1 - 1/e$-approximate [Nguyen et al., 2013; Krause et al., 2008; Nemhauser et al., 1978].

4.1.2 Approximating $a_{\texttt{BatchBALD}}$ via *Consistent* Monte-Carlo Sampling

For brevity, we leave out conditioning on $\boldsymbol{x}_{1..n}$, and $\mathcal{D}^{\text{train}}$, and $\mathrm{p}(\boldsymbol{\omega})$ denotes $\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$ in this section. $a_{\texttt{BatchBALD}}$ is then written as:

$$a_{\texttt{BatchBALD}}\left(\{\boldsymbol{x}_{1..n}\}, \mathrm{p}(\boldsymbol{\omega})\right) = \mathrm{H}[Y_{1..n}] - \mathrm{H}[Y_{1..n} \mid \boldsymbol{\Omega}] \tag{4.5}$$
$$= \mathrm{H}[Y_{1..n}] - \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}[\mathrm{H}[Y_{1..n} \mid \boldsymbol{\omega}]]. \tag{4.6}$$

Because the Y_i are independent when conditioned on $\boldsymbol{\omega}$, computing the right term of equation (4.5) is simplified as the conditional joint entropy decomposes into a sum. We can approximate the expectation using a Monte-Carlo estimator with k samples from our model parameter distribution $\hat{\boldsymbol{\omega}}_j \sim \mathrm{p}(\boldsymbol{\omega})$:

$$\mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}[\mathrm{H}[Y_{1..n} \mid \boldsymbol{\omega}]] = \sum_{i=1}^{n} \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}[\mathrm{H}[Y_i \mid \boldsymbol{\omega}]] \approx \frac{1}{k} \sum_{i=1}^{n} \sum_{j=1}^{k} \mathrm{H}[Y_i \mid \hat{\boldsymbol{\omega}}_j]. \tag{4.7}$$

Crucially, the samples have to stay fixed across different pool samples. This is *not* how Monte-Carlo dropout [Gal and Ghahramani, 2016a] is usually implemented. We call this *consistent* Monte-Carlo dropout. See Figure 4.5 for why this is necessary (based on real data).

Computing the left term of equation (4.5) is difficult because the unconditioned joint probability does not factorize. Applying the equality $\mathrm{p}(y) = \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}[\mathrm{p}(y \mid \boldsymbol{\omega})]$, and, using sampled $\hat{\boldsymbol{\omega}}_j$, we compute the entropy by summing over all possible configurations $\hat{y}_{1:n}$ of $y_{1:n}$:

$$\mathrm{H}[Y_{1..n}] = \mathbb{E}_{\mathrm{p}(y_{1..n})}[-\log \mathrm{p}(y_{1..n})] \tag{4.8}$$
$$= \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}\, \mathbb{E}_{\mathrm{p}(y_{1..n} \mid \boldsymbol{\omega})}[-\log \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}[\mathrm{p}(y_{1..n} \mid \boldsymbol{\omega})]] \tag{4.9}$$
$$\approx -\sum_{\hat{y}_{1:n}} \left(\frac{1}{k} \sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{\omega}}_j) \right) \log \left(\frac{1}{k} \sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{\omega}}_j) \right). \tag{4.10}$$

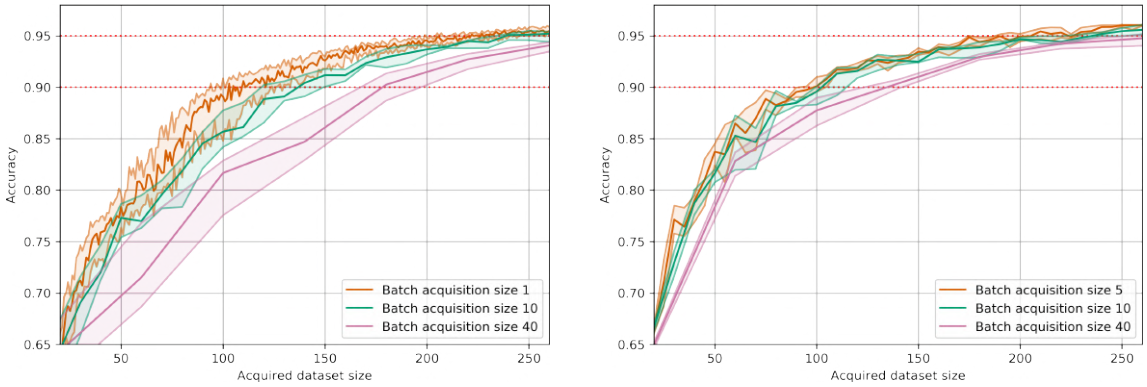
4.1.3 Efficient Estimation

In each iteration of the algorithm, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$ stay fixed while $\boldsymbol{x}_n$ varies over $\mathcal{D}^{\text{pool}} \setminus A_{n-1}$. We can reduce the required computations by factorizing $\mathrm{p}(y_{1:n} \mid \boldsymbol{\omega})$ into $\mathrm{p}(y_{1:n-1} \mid \boldsymbol{\omega})\, \mathrm{p}(y_n \mid \boldsymbol{\omega})$. We store $\mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{\omega}}_j)$ in a matrix $\hat{P}_{1:n-1}$ of shape $\mathtt{C}^{n-1} \times k$ and $\mathrm{p}(y_n \mid \hat{\boldsymbol{\omega}}_j)$ in a matrix $\hat{P}_n$ of shape $\mathtt{C} \times k$. The sum $\sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{\omega}}_j)$ in (4.10) can be then be turned into a matrix product:

$$\frac{1}{k} \sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{\omega}}_j) = \frac{1}{k} \sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{\omega}}_j)\, \mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{\omega}}_j) = \left(\frac{1}{k} \hat{P}_{1:n-1} \hat{P}_n^T \right)_{\hat{y}_{1:n-1}, \hat{y}_n}. \tag{4.11}$$

This can be further sped up by using batch matrix multiplication to compute the joint entropy for different $\boldsymbol{x}_n$. $\hat{P}_{1:n-1}$ only has to be computed once, and we can recursively compute $\hat{P}_{1:n}$ using $\hat{P}_{1:n-1}$ and $\hat{P}_n$, which allows us to sample $\mathrm{p}(y \mid \hat{\boldsymbol{\omega}}_j)$ for each $\boldsymbol{x} \in \mathcal{D}^{\text{pool}}$ only once at the beginning of the algorithm.

For larger acquisition sizes, we use m MC samples of $y_{1:n-1}$ as enumerating all possible configurations becomes infeasible. See Appendix D.3 for details.



(a) BALD **(b) BatchBALD**

Figure 4.6: *Performance on* MNIST *for increasing acquisition sizes.* BALD's performance drops drastically as the acquisition size increases. BatchBALD maintains strong performance even with increasing acquisition size.

Monte-Carlo sampling bounds the time complexity of the full BatchBALD algorithm to $\mathcal{O}(\mathtt{KC} \cdot \min\{\mathtt{C}^{\mathtt{K}}, m\} \cdot |\mathcal{D}^{\text{pool}}| \cdot k)$ compared to $\mathcal{O}(\mathtt{C}^{\mathtt{K}} \cdot |\mathcal{D}^{\text{pool}}|^{\mathtt{K}} \cdot k)$ for naively finding the exact optimal batch and $\mathcal{O}((\mathtt{K}+k) \cdot |\mathcal{D}^{\text{pool}}|)$ for BALD[2].

4.2 Empirical Validation

In our experiments, we start by showing how a naive application of the BALD algorithm to an image dataset can lead to poor results in a dataset with many (near) duplicate data points and show that BatchBALD solves this problem in a grounded way while obtaining favorable results (Figure 4.2). We then illustrate BatchBALD's effectiveness on standard active learning datasets: MNIST and EMNIST. EMNIST [Cohen et al., 2017] is an extension of MNIST that also includes letters, for a total of 47 classes and has twice as large a training set. See Appendix D.6 for examples of the dataset. We show that BatchBALD provides a substantial performance improvement in these scenarios, too, and has more diverse acquisitions. Finally, we look at BatchBALD in the setting of transfer learning, where we fine-tune a large pretrained model on a more difficult dataset called CINIC-10 [Darlow et al., 2018], which is a combination of CIFAR-10 and down-scaled ImageNet.

In our experiments, we repeatedly go through active learning loops. One active learning loop consists of training the model on the available labeled data and subsequently acquiring new data points using a chosen acquisition function. As the labeled dataset is small in the beginning, it is important to avoid overfitting. We do this by using early stopping after 3 epochs of declining accuracy on the validation set. We pick the model with the highest validation accuracy. Throughout our experiments, we use the Adam [Kingma and Ba, 2015] optimizer with learning rate 0.001 and betas 0.9/0.999. All our results report the median of 6 trials, with lower and upper quartiles. We use these quartiles to draw the filled error bars on our figures.

We reinitialize the model after each acquisition, similar to Gal et al. [2017]: empirically, we found this helps the model improve even when very small batches

[2] $\mathtt{K}$ is the acquisition size, $\mathtt{C}$ is the number of classes, k is the number of MC dropout samples, and m is the number of sampled configurations of $y_{1:n-1}$.

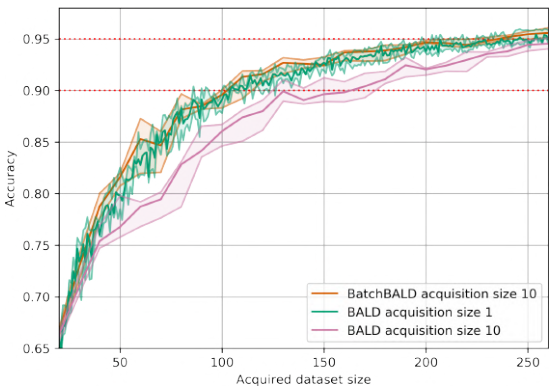


Figure 4.7: *Performance on* MNIST. BatchBALD outperforms BALD with acquisition size 10 and performs close to the optimum of acquisition size 1.

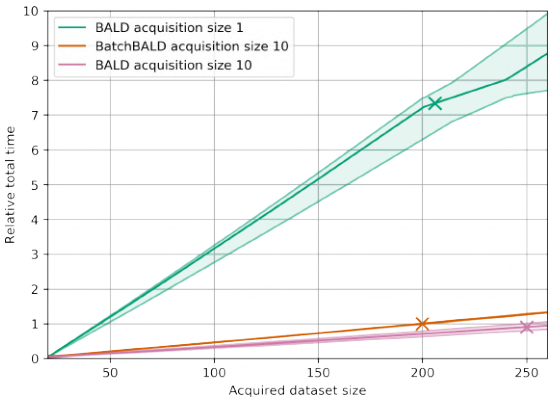


Figure 4.8: *Relative total time on* MNIST. Normalized to training BatchBALD with acquisition size 10 to 95% accuracy. The stars mark when 95% accuracy is reached for each method.

are acquired. It also decorrelates subsequent acquisitions as final model performance is dependent on a particular initialization [Frankle and Carbin, 2019].

When computing $\mathrm{p}(y \mid \boldsymbol{x}, \boldsymbol{\omega}, \mathcal{D}^{\text{train}})$, it is important to keep the dropout masks in MC dropout consistent while sampling from the model. This is necessary to capture dependencies between the inputs for BatchBALD, and it makes the scores for different points more comparable by removing this source of noise. We do not keep the masks fixed when computing BALD scores because its performance usually benefits from the added noise. We also do not need to keep these masks fixed for training and evaluating the model.

In all our experiments, we either compute joint entropies exactly by enumerating all configurations, or we estimate them using 10,000 MC samples, picking whichever method is faster. In practice, we compute joint entropies exactly for roughly the first 4 data points in an acquisition batch and use MC sampling thereafter.

4.2.1 Repeated-MNIST

As demonstrated in the introduction, naively applying BALD to a dataset that contains many (near) replicated data points leads to poor performance. We show how this manifests in practice by taking the MNIST dataset and replicating each data point in the training set twice (obtaining a training set that is three times larger than the original). After normalizing the dataset, we add isotropic Gaussian noise with a standard deviation of 0.1 to simulate slight differences between the duplicated data points in the training set. All results are obtained using an acquisition size of 10 and 10 MC dropout samples. The initial dataset was constructed by taking a balanced set of 20 data points[3], two of each class, similar to [Gal et al., 2017].

Our model consists of two blocks of [convolution, dropout, max-pooling, relu], with 32 and 64 5x5 convolution filters. These blocks are followed by a two-layer MLP that includes dropout between the layers and has 128 and 10 hidden units. The dropout probability is 0.5 in all three locations. This architecture achieves 99% accuracy with

[3]These initial data points were chosen by running BALD 6 times with the initial dataset picked randomly and choosing the set of the median model. They were subsequently held fixed.

Table 4.1: *Number of required data points on* MNIST *until 90% and 95% accuracy are reached.* 25%-, 50%- and 75%-quartiles for the number of required data points when available.

Accuracy	**90%**	**95%**
BatchBALD	70 / 90 / 110	190 / 200 / 230
BALD [4]	120 / 120 / 170	250 / 250 / >300
BALD [Gal et al., 2017]	145	335

10 MC dropout samples during test time on the full MNIST dataset.

The results can be seen in Figure 4.2. In this illustrative scenario, BALD performs poorly, and even randomly acquiring points performs better. However, BatchBALD is able to cope with the replication perfectly. In Appendix D.4, we look at varying the repetition number and show that as we increase the number of repetitions BALD gradually performs worse. In Appendix D.5, we also compare with Variation Ratios [Freeman, 1965], and Mean STD [Kendall et al., 2017] which perform on par with random acquisition.

4.2.2 MNIST

For the second experiment, we follow the setup of Gal et al. [2017] and perform active learning on the MNIST dataset using 100 MC dropout samples. We use the same model architecture and initial dataset as described in section 4.2.1. Due to differences in model architecture, hyperparameters and model retraining, we significantly outperform the original results in Gal et al. [2017] as shown in table 4.1.

We first look at BALD for increasing acquisition size in Figure 4.6(a). As we increase the acquisition size from the ideal of acquiring points individually and fully retraining after each point (acquisition size 1) to 40, there is a substantial performance drop.

BatchBALD, in Figure 4.6(b), is able to maintain performance when doubling the acquisition size from 5 to 10. Performance drops only slightly at 40, possibly due to estimator noise.

The results for acquisition size 10 for both BALD and BatchBALD are compared in Figure 4.7. BatchBALD outperforms BALD. Indeed, BatchBALD with acquisition size 10 performs close to the ideal with acquisition size 1. The total run time of training these three models until 95% accuracy is visualized in Figure 4.8, where we see that BatchBALD with acquisition size 10 is much faster than BALD with acquisition size 1, and only marginally slower than BALD with acquisition size 10.

4.2.3 EMNIST

In this experiment, we show that BatchBALD also provides a significant improvement when we consider the more difficult EMNIST dataset [Cohen et al., 2017] in the *Balanced* setup, which consists of 47 classes, comprising letters and digits. The training set consists of 112,800 28x28 images balanced by class, of which the last 18,800 images constitute the validation set. We do not use an initial dataset and

[4] reimplementation using reported experimental setup

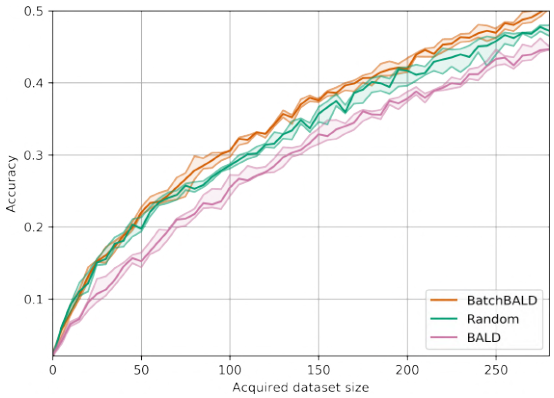


Figure 4.9: *Performance on* EMNIST. BatchBALD consistently outperforms both random acquisition and BALD while BALD is unable to beat random acquisition.

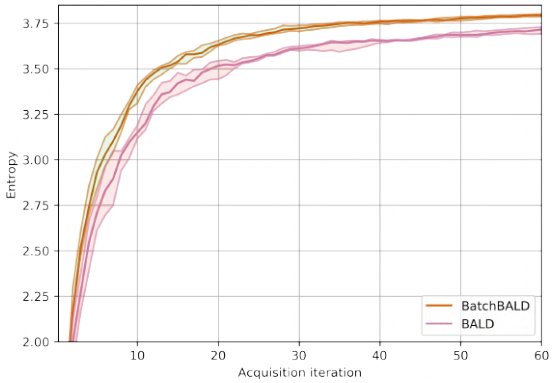


Figure 4.10: *Entropy of acquired class labels over acquisition steps on* EMNIST. BatchBALD steadily acquires a more diverse set of data points.

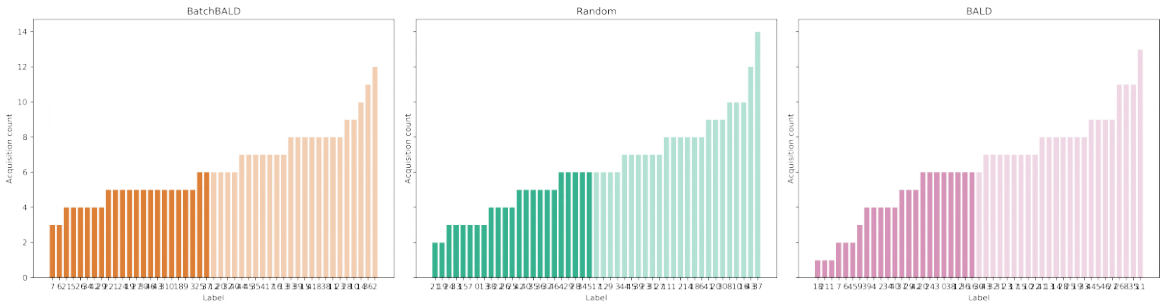


Figure 4.11: *Histogram of acquired class labels on* EMNIST. BatchBALD left and BALD right. Classes are sorted by number of acquisitions. Several EMNIST classes are underrepresented in BALD and random acquisition while BatchBALD acquires classes more uniformly. The histograms were created from all acquired points at the end of an active learning loop

instead perform the initial acquisition step with the randomly initialized model and use 10 MC dropout samples.

We use a similar model architecture as before, but with added capacity. Three blocks of [convolution, dropout, max-pooling, relu], with 32, 64 and 128 3x3 convolution filters, and 2x2 max pooling. These blocks are followed by a two-layer MLP with 512 and 47 hidden units, with again a dropout layer in between. We use dropout probability 0.5 throughout the model.

The results for acquisition size 5 can be seen in Figure 4.9. BatchBALD outperforms both random acquisition and BALD while BALD is unable to beat random acquisition. Figure 4.10 gives some insight into why BatchBALD performs better than BALD. The entropy of the categorical distribution of acquired class labels is consistently higher, meaning that BatchBALD acquires a more diverse set of data points. In Figure 4.11, the classes on the x-axis are sorted by number of data points that were acquired of that class. We see that BALD undersamples classes while BatchBALD is more consistent.

4.2.4 CINIC-10

CINIC-10 is an interesting dataset because it is large (270k data points) and its data comes from two different sources: CIFAR-10 and ImageNet. To get strong performance

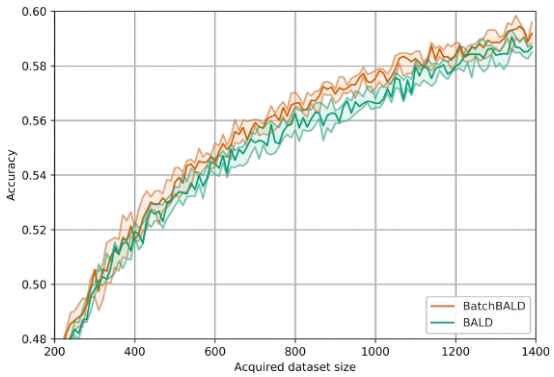


Figure 4.12: *Performance on* CINIC-10. BatchBALD outperforms BALD from 500 acquired samples onwards.

on the test set it is important to obtain data from both sets. Instead of training a very deep model from scratch on a small dataset, we opt to run this experiment in a transfer learning setting, where we use a pretrained model and acquire data only to fine-tune the original model. This is common practice and suitable in cases where there is plenty of data available in an auxiliary domain, but it is expensive to label data for the domain of interest.

For the CINIC-10 experiment, we use 160k training samples for the unlabeled pool, 20k validation samples, and the other 90k as test samples. We use an ImageNet pretrained VGG-16, provided by PyTorch [Paszke et al., 2017], with a dropout layer before a 512 hidden unit (instead of 4096) fully connected layer. We use 50 MC dropout samples, acquisition size 10 and repeat the experiment for 6 trials. The results are in Figure 4.12, with the 59% mark reached at 1170 for BatchBALD and 1330 for BALD (median).

4.3 Discussion

We have introduced a new batch acquisition function, BatchBALD, for Deep Bayesian Active Learning, and a greedy algorithm that selects good candidate batches compared to the intractable optimal solution. Acquisitions show increased diversity of data points and improved performance over BALD and other methods.

While our method comes with additional computational cost during acquisition, BatchBALD is able to significantly reduce the number of data points that need to be labeled and the number of times the model has to be retrained, potentially saving considerable costs and filling an important gap in practical Deep Bayesian Active Learning.

This proposed method of batch acquisition has some weaknesses and limitations, which we discuss here.

- **Unbalanced Datasets.** BALD and BatchBALD do not work well when the test set is unbalanced as they aim to learn well about all classes and do not follow the density of the dataset. However, if the test set is balanced, but the training set is not, we expect BatchBALD to perform well.
- **Unlabeled Data.** BatchBALD does not take into account any information from the unlabeled dataset. However, BatchBALD uses the underlying Bayesian model for estimating uncertainty for unlabeled data points, and semi-supervised learning could improve these estimates by providing more information about the

underlying structure of the feature space. We leave a semi-supervised extension of BatchBALD to future work.

- **Noisy Estimator.** A significant amount of noise is introduced by MC-dropout's variational approximation to training BNNs. Sampling of the joint entropies introduces additional noise. The quality of larger acquisition batches would be improved by reducing this noise. We examine this further in §5 and 6.

I speak without a mouth and hear without ears.
I have no body, but I come alive with the wind.
What am I?

5

Stochastic Batch Acquisition for Deep Active Learning

While many acquisition schemes are designed to acquire labels one at a time [Houlsby et al., 2011; Gal et al., 2017], we have already highlighted the importance of *batch acquisition* in §1.2.4 and §4. Unfortunately, existing batch acquisition schemes are computationally expensive (Table 5.1). Intuitively, this is because batch acquisition schemes face combinatorial complexity when accounting for the interactions between possible acquisition points. Recent works [Ash et al., 2020, 2021] trade off a principled motivation with various approximations to remain tractable. A commonly used, though extreme, heuristic is to take the top-$\mathtt{K}$ highest scoring points from an acquisition scheme designed to select a single point.

This chapter introduces a simple baseline for batch active learning that can be competitive with methods that cost orders of magnitude more across a wide range of experimental contexts. The presented method is motivated by noticing that single-acquisition score methods such as BALD [Houlsby et al., 2011] act as a noisy proxy for future acquisition scores (Figure 5.1). This observation leads us to stochastically acquire points following a distribution determined by the single-acquisition scores. Importantly, such a simple approach can match the prior state of the art for batch acquisition despite being very simple. Moreover, this acquisition scheme has a time complexity of only $\mathcal{O}(\mathtt{M} \log \mathtt{K})$ in the pool size $\mathtt{M}$ and acquisition size $\mathtt{K}$, just like top-$\mathtt{K}$ acquisition.

We show empirically that the presented stochastic strategy performs as well as or better than top-$\mathtt{K}$ acquisition with almost identical computational cost on several commonly used acquisition scores, making it a strictly-better batch strategy. Strikingly, the empirical comparisons between this stochastic strategy and the evaluated more complex methods cast doubt on whether they function as well as claimed. Concretely, in this chapter, we:

- examine a family of three computationally cheap stochastic batch acquisition strategies;
- demonstrate that these strategies are preferable to the commonly used top-$\mathtt{K}$ acquisition heuristic; and
- identify the failure of existing SotA batch acquisition strategies to outperform this vastly cheaper and more heuristic strategy.

In §5.1, we present active learning notation and commonly used acquisition functions. We propose stochastic extensions in §5.2, relate them to previous work in §5.3, and validate them empirically in §5.4 on various datasets, showing that these extensions are competitive with some much more complex active learning approaches despite being

Table 5.1: *Acquisition runtime (in seconds, 5 trials, ± s.d.).* The examined stochastic acquisition methods are as fast as top-K, and **orders of magnitude** faster than BADGE or BatchBALD. Synthetic pool set with $M = 10,000$ pool points with 10 classes. BatchBALD and BALD with 20 parameter samples.

K	Top-K	**Stochastic**	BADGE	BatchBALD
10	0.2 ± 0.0	0.2 ± 0.0	9.2 ± 0.3	566.0 ± 17.4
100	0.2 ± 0.0	0.2 ± 0.0	82.1 ± 2.5	$5,363.6 \pm 95.4$
500	0.2 ± 0.0	0.2 ± 0.0	409.3 ± 3.7	$29,984.1 \pm 598.7$

orders of magnitude computationally cheaper. Finally, we validate the underlying theoretical motivation in §5.5 and discuss limitations in §5.6.

5.1 Problem Setting

The stochastic approach we examine applies to batch acquisition for active learning in a pool-based setting [Settles, 2010] where we have access to a large unlabeled *pool* set, but we can only label a small subset of the points. The challenge of active learning is to use what we already know to pick which points to label in the most efficient way, and generally, we want to avoid labelling points similar to those already labeled.

Notation. Following Farquhar et al. [2021], and unlike in the rest of the thesis, we formulate active learning over *indices* instead over data points. This simplifies the notation. The large, initially fully unlabeled, pool set containing $\mathtt{M}$ input points is

$$\mathcal{D}^{\text{pool}} = \{x_i\}_{i \in \mathcal{I}^{\text{pool}}}, \tag{5.1}$$

where $\mathcal{I}^{\text{pool}} = \{1, \ldots, \mathtt{M}\}$ is the initial full index set. We initialize a training dataset with $\mathtt{N}_0$ randomly selected points from $\mathcal{D}^{\text{pool}}$ by acquiring their labels, y_i,

$$\mathcal{D}^{\text{train}} = \{(x_i, y_i)\}_{i \in \mathcal{I}^{\text{train}}}, \tag{5.2}$$

where $\mathcal{I}^{\text{train}}$ is the index set of $\mathcal{D}^{\text{train}}$, *initially* containing $\mathtt{N}_0$ indices between 1 and $\mathtt{M}$. A model of the predictive distribution, $\mathrm{p}(y \mid \boldsymbol{x})$, can then be trained on $\mathcal{D}^{\text{train}}$.

Active Learning. At each acquisition step, we select additional points for which to acquire labels. Although many methods acquire one point at a time [Houlsby et al., 2011; Gal et al., 2017], one can alternatively acquire a whole batch of $\mathtt{K}$ examples. An acquisition function a takes $\mathcal{I}^{\text{train}}$ and $\mathcal{I}^{\text{pool}}$ and returns $\mathtt{K}$ indices from $\mathcal{I}^{\text{pool}}$ to be added to $\mathcal{I}^{\text{train}}$. We then label those $\mathtt{K}$ data points and add them to $\mathcal{I}^{\text{train}}$ while making them unavailable from the pool set. That is,

$$\mathcal{I}^{\text{train}} \leftarrow \mathcal{I}^{\text{train}} \cup a(\mathcal{I}^{\text{train}}, \mathcal{I}^{\text{pool}}), \tag{5.3}$$

$$\mathcal{I}^{\text{pool}} \leftarrow \mathcal{I}^{\text{pool}} \setminus \mathcal{I}^{\text{train}}. \tag{5.4}$$

A common way to construct the acquisition function is to define some scoring function, s, and then select the point(s) that score the highest.

We consider the following scoring functions:

BALD. For each candidate pool index, i, the BALD score is

$$\begin{aligned} s_{\text{BALD}}(i; \mathcal{I}^{\text{train}}) &\triangleq \mathrm{I}[Y; \Omega \mid X = x_i, \mathcal{D}^{\text{train}}] \\ &= \mathrm{H}[Y \mid X = x_i, \mathcal{D}^{\text{train}}] - \mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})}[\mathrm{H}[Y \mid X = x_i, \omega, \mathcal{D}^{\text{train}}]]. \end{aligned} \tag{5.5}$$

Entropy. Entropy does not require Bayesian models, unlike BALD, and performs worse for data with high observation noise as we have noted in §3. It is identical to the first term of the BALD score

$$s_{\text{entropy}}(i; \mathcal{I}^{\text{train}}) \triangleq \mathrm{H}[Y \mid X = x_i, \mathcal{D}^{\text{train}}]. \tag{5.6}$$

See §1.2.4 for other acquisition functions.

Acquisition Functions. These scoring functions were introduced for single-point acquisition:

$$a_s(\mathcal{I}^{\text{train}}) \triangleq \operatorname*{arg\,max}_{i \in \mathcal{I}^{\text{pool}}} s(i; \mathcal{I}^{\text{train}}). \tag{5.7}$$

For deep learning in particular, single-point acquisition is computationally expensive due to retraining the model for every acquired sample. Moreover, it also means that labelling can only happen sequentially instead of in bulk. Thus, single-point acquisition functions were expanded to multi-point acquisition via acquisition batches in batch active learning. The most naive batch acquisition function selects the highest `K` scoring points

$$a_s^{\text{batch}}(\mathcal{I}^{\text{train}}; \mathtt{K}) \triangleq \operatorname*{arg\,max}_{I \subseteq \mathcal{I}^{\text{pool}}, |I| = \mathtt{K}} \sum_{i \in I} s(i; \mathcal{I}^{\text{train}}). \tag{5.8}$$

Maximizing this sum is equivalent to taking the top-`K` scoring points, which cannot account for the interactions between points in an acquisition batch because individual points are scored independently. Some acquisition functions are explicitly designed for batch acquisition, e.g. BatchBALD from §4 or BADGE from Ash et al. [2020]. They try to account for the interaction between points, which can improve performance relative to simply selecting the top-`K` scoring points. However, existing methods are computationally expensive. For example, BatchBALD rarely scales to acquisition sizes of more than 5–10 points as noted in §4; see Table 5.1.

BADGE. Ash et al. [2020] propose *Batch Active learning by Diverse Gradient Embeddings*: it motivates its batch selection approach using a k-Determinantal Point Process [Kulesza and Taskar, 2011] based on the (inner product) similarity matrix of the scores (gradients of the log loss) using hard pseudo-labels (the highest probability class according to the model's prediction) for each pool sample. In §9 we provide a more detailed analysis. In practice, they use the initialization step of k-MEANS++ with Euclidian distances between the scores to select an acquisition batch. BADGE is also computationally expensive.

5.2 Method

We observe that selecting the top-`K` points at acquisition step t amounts to the assumption that the informativeness of these points is independent of each other. Imagine adding the top-`K` points at a given acquisition step t to the training set one

Table 5.2: *Summary of stochastic acquisition variants.* Perturbing the scores s_i themselves with $\epsilon_i \sim \text{Gumbel}(0; \beta^{-1})$ i.i.d. yields a softmax distribution. Log-scores result in a power distribution, with assumptions that are reasonable for active learning. Using the score-ranking, r_i finally is a robustifying assumption. β is included for completeness; we use $\beta \triangleq 1$ in our experiments—except for the ablation in §5.5.1.

Perturbation	Distribution	Probability mass
$s_i + \epsilon_i$	Softmax	$\propto \exp \beta s_i$
$\log s_i + \epsilon_i$	Power	$\propto s_i^{\beta}$
$-\log r_i + \epsilon_i$	Soft-rank	$\propto r_i^{-\beta}$

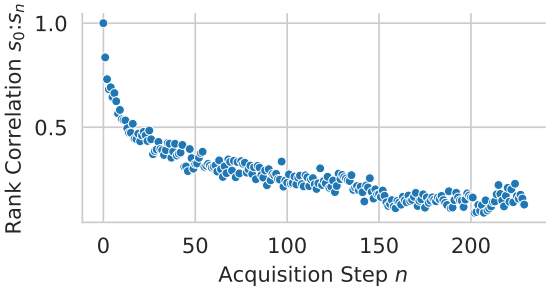


Figure 5.1: *Early acquisition scores are only a loose proxy for later scores.* Specifically, the Spearman rank-correlation between acquisition scores on the first and n'th time-step falls with n. While top-K acquisition incorrectly implicitly assumes the rank-correlation remains 1, stochastic acquisitions do not. BNN trained on MNIST at initial 20 points and 73% initial accuracy, score ranks over test set.

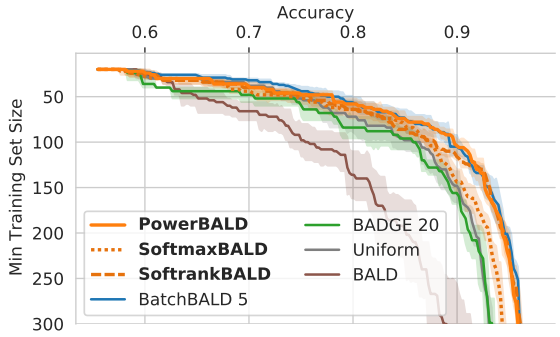


Figure 5.2: *Performance on Repeated-MNIST with 4 repetitions (5 trials).* **Up and to the right is better (↗).** PowerBALD outperforms (top-K) BALD and BADGE and is on par with BatchBALD. This is despite being orders of magnitude faster. Acquisition sizes: BatchBALD–5, BADGE–20, others–10. See Figure E.2 in the appendix for an ablation study of BADGE's acquisition size.

at a time. Each time, you retrain the model. Of course, the acquisition scores for the models trained with these additional points will be different from the first set of scores. After all, the purpose of active learning is to add the *most informative* points: those that will update the model the most. Yet selecting a top-K batch in one step implicitly assumes that the score ranking will not change due to these points. This is clearly wrong. We provide empirical confirmation that, in fact, the ranking of acquisition scores at step t and $t+\texttt{K}$ is decreasingly correlated as K grows; see Figure 5.1. Moreover, this effect is the strongest for the most informative points; see §5.5 for more details.

Instead, this chapter uses stochastic sampling to acknowledge the uncertainty within the batch acquisition step using a simple noise process model governing how scores change. We examine three simple stochastic extensions of single-sample scoring functions $s(i; \mathcal{I}^{\text{train}})$ that make slightly different assumptions. These methods are compatible with conventional active learning frameworks that typically take the top-K highest scoring samples. For example, it is straightforward to adapt entropy, BALD, and other scoring functions for use with these extensions.

These stochastic acquisition distributions assume that future scores differ from the current score by a perturbation. We model the noise distribution of this perturbation as the addition of Gumbel-distributed noise $\epsilon_i \sim \text{Gumbel}(0; 1)$, which is used frequently for modelling extrema. The choice of a Gumbel distribution for the noise is one of mathematical convenience, in the spirit of providing a simple baseline. For example, the maximum of sets of many other standard distributions, such as the Gaussian distribution, is not analytically tractable.

Taking the highest-scoring points from this perturbed distribution is equivalent to sampling from a softmax distribution[1] without replacement with a 'coldness' parameter $\beta \geq 0$, which represents the expected rate at which the scores change as more data is acquired. This follows from the Gumbel-Max trick [Gumbel, 1954; Maddison et al., 2014] and, more specifically, the Gumbel-Top-K trick [Kool et al., 2019]. We provide a short proof in appendix E.1. Expanding on Maddison et al. [2014]:

Proposition 5.1. *For scores s_i, $i \in \{1, \ldots, n\}$, and $k \leq n$ and $\beta > 0$, if we draw $\epsilon_i \sim$ Gumbel$(0; \beta^{-1})$ independently, then* $\arg\operatorname{top}_k\{s_i + \epsilon_i\}_i$ *is an (ordered) sample without replacement from the categorical distribution* $\text{Categorical}(\exp(\beta s_i)/\sum_j \exp(\beta s_j), i \in \{1, \ldots, n\})$.

In the spirit of providing a simple and surprisingly effective baseline without hyperparameters, we fix $\beta \triangleq 1$. For $\beta \to \infty$, this distribution will converge towards top-K acquisition. Whereas for $\beta \to 0$, it will converge towards uniform acquisition. We examine ablations of β in §5.5.1.

We apply the perturbation to three quantities in the three sampling schemes: the scores themselves, the log scores, and the rank of the scores. Perturbing the log scores assumes that scores are non-negative and uninformative points should be avoided. Perturbing the ranks can be seen as a robustifying assumption that requires the relative scores to be reliable but allows the absolute scores to be unreliable. We summarize the three versions with their associated sampling distributions are in Table 5.2.

Soft-Rank Acquisition. This first variant only relies on the rank order of the scores and makes no assumptions on whether the acquisition scores are meaningful beyond that. It thus uses the *least* amount of information from the acquisition scores. It only requires the *relative score order* to be useful and ignores the *absolute score values*. If the absolute scores provide useful information, we would expect this method to perform worse than the variants below, which make use of the score values. As we will see, this is indeed sometimes the case.

Ranking the scores $s(i; \mathcal{I}^{\text{train}})$ with descending ranks $\{r_i\}_{i \in \mathcal{I}^{\text{pool}}}$ such that $s(r_i; \mathcal{I}^{\text{train}}) \geq s(r_j; \mathcal{I}^{\text{train}})$ for $r_i \leq r_j$ and smallest rank being 1, we sample index i with probability $\text{p}_{\text{softrank}}(i) \propto r_i^{-\beta}$ with coldness β. This is invariant to the actual scores. We can draw $\epsilon_i \sim \text{Gumbel}(0; \beta^{-1})$ and create a perturbed 'rank'

$$s^{\text{softrank}}(i; \mathcal{I}^{\text{train}}) := -\log r_i + \epsilon_i. \tag{5.9}$$

Taking the top-K samples is now equivalent to sampling without replacement from the rank distribution $\text{p}_{\text{softrank}}(i)$.

[1] Also known as Boltzmann/Gibbs distribution.

Softmax Acquisition. The next simplest variant uses the actual scores instead of the ranks. Again, it perturbs the scores by a Gumbel-distributed random variable $\epsilon_i \sim \text{Gumbel}(0; \beta^{-1})$

$$s^{\text{softmax}}(i; \mathcal{I}^{\text{train}}) := s(i; \mathcal{I}^{\text{train}}) + \epsilon_i. \tag{5.10}$$

However, this makes no assumptions about the semantics of the absolute values of the scores: the softmax function is invariant to constants shifts. Hence, the sampling distribution will only depend on the *relative scores* and not their absolute value.

Power Acquisition. For many scoring functions, the scores are non-negative, and a score close to zero means that the sample is not informative in the sense that we do not expect it will improve the model—we do not want to sample it. This is the case with commonly used score functions such as BALD and entropy. BALD measures the expected information gain. When it is zero for a sample, we do not expect anything to be gained from acquiring a label for that sample. Similarly, entropy is upper-bounding BALD, and the same consideration applies. This assumption also holds ideally for other scoring functions that are easily transformed to be non-negative (see §1.2.4). To take this into account, the last variant models the future log scores as perturbations of the current log score with Gumbel-distributed noise

$$s^{\text{power}}(i; \mathcal{I}^{\text{train}}) := \log s(i; \mathcal{I}^{\text{train}}) + \epsilon_i. \tag{5.11}$$

By Proposition 5.1, this is equivalent to sampling from a power distribution

$$\mathrm{p}_{power}(i) \propto \left(\frac{1}{s(i; \mathcal{I}^{\text{train}})} \right)^{-\beta}. \tag{5.12}$$

This may be seen by noting that $\exp(\beta \log s(i; \mathcal{I}^{\text{train}})) = s(i; \mathcal{I}^{\text{train}})^{\beta}$. Importantly, as scores $\to 0$, the (perturbed) log scores $\to -\infty$ and will have probability mass $\to 0$ assigned. This variant takes the absolute scores into account and avoids data points with score 0.

Summary. Given the above considerations, when using BALD, entropy, and other appropriate scoring functions, power acquisition is the most sensible. Thus, we expect it to work best. Indeed, we find this to be the case in the toy experiment on Repeated-MNIST (see §4) depicted in Figure 5.2. However, even soft-rank acquisition works well in practice, suggesting that the choice of score perturbation is not critical for its effectiveness; see also appendix §E.3 for a more in-depth comparison. In the rest of this chapter, we focus on power acquisition—we include results for all methods in §E.2.

5.3 Related Work

In most cases, the computational complexity scales poorly with the acquisition size ($\mathtt{K}$) or pool size ($\mathtt{M}$), for example because of the estimation of joint mutual information (§4), the $\mathcal{O}(\mathtt{KM})$ complexity of using a k-means++ initialization scheme [Ash et al., 2020], which approximates k-DPP-based batch active learning [Bıyık et al., 2019], or the $\mathcal{O}(\mathtt{M}^2 \log \mathtt{M})$ complexity of methods based on $\mathtt{K}$-center coresets [Sener and Savarese, 2018] (although heuristics and continuous relaxations can improve this somewhat). In contrast, we examine simple and efficient stochastic strategies for adapting well-known

single-sample acquisition functions to the batch setting. The proposed stochastic strategies are based on observing that acquisition scores would change as new points are added to the acquisition batch and modelling this difference for additional batch samples in the most naive way, using Gumbel noise. The presented stochastic extensions have the same complexity $\mathcal{O}(\mathtt{M} \log \mathtt{K})$ as naive top-K batch acquisition, yet outperform it, and they can perform on par with above more complex methods.

For multi-armed bandits, it has been shown that Thompson sampling from the posterior is effective for choosing informative batches [Kalkanli and Özgür, 2021]. Compared to using the Bayesian model average of the posterior, this can be seen as noising the BMA acquisition scores. Similarly, in reinforcement learning, stochastic prioritization has been employed as *prioritized replay* [Schaul et al., 2016] which may be effective for reasons analogous to those motivating the approach examined in this chapter.

While stochastic sampling has not been extensively explored for acquisition in deep active learning, most recently it has been used as an auxiliary step in diversity-based active learning methods that rely on clustering as main mechanism [Ash et al., 2020; Citovsky et al., 2021]. In §4 we noted that additional noise in the acquisition scores seems to benefit top-K batch acquisition in our experiments but did not investigate further. Fredlund et al. [2010] suggest modeling single-point acquisition as sampling from a "*query density*" modulated by the (unknown) sample density $\mathrm{p}(x)$ and analyze a binary classification toy problem. They model the query density using a parameterized model. Farquhar et al. [2021] propose stochastic acquisition as part of debiasing actively learned estimators.

Most relevant to this chapter, and building on Fredlund et al. [2010]; Farquhar et al. [2021], Zhan et al. [2022b] propose a stochastic acquisition scheme that is asymptotically optimal. They normalize the acquisition scores via the softmax function to obtain a query density function for unlabeled samples and draw an acquisition batch from it, similar to SoftmaxEntropy. Their method aims to achieve asymptotic optimality for active learning processes by mitigating the impact of bias. However, in this chapter, we propose multiple stochastic acquisition strategies based on score-based or rank-based distributions and apply these strategies to several single-sample acquisition functions, such as BALD and entropy (and standard deviation, variation ratios, see Figure E.3). We focus on active learning in a (Bayesian) deep learning setting and not in a classical machine learning setting. As such the empirical results and other proposed strategies can be seen as complementary to their work.

Thus, while stochastic sampling is generally well-known within acquisition functions, entirely simple stochastic sampling have not been investigated as alternatives to naive top-K acquisition in (Bayesian) deep active learning and compared to more complex approaches in various settings.

5.4 Empirical Validation

In this section, we empirically verify that the presented stochastic acquisition methods (a) outperform top-K acquisition and (b) are competitive with specially designed batch acquisition schemes like BADGE [Ash et al., 2020] and BatchBALD (§4); and are vastly cheaper than these more complicated methods.

To demonstrate the seriousness of the possible weakness of recent batch acquisition methods, we use a range of datasets. These experiments show that the performance of the stochastic extensions is not dependent on the specific characteristics of any particular

dataset. Our experiments include computer vision, natural language processing (NLP), and causal inference (in §5.5.1). We show that stochastic acquisition helps avoid selecting redundant samples on Repeated-MNIST (§4), examine performance in active learning for computer vision on EMNIST [Cohen et al., 2017], MIO-TCD [Luo et al., 2018], Synbols [Lacoste et al., 2020], and CLINC-150 [Larson et al., 2019] for intent classification in NLP. MIO-TCD is especially close to real-world datasets in size and quality. In appendix E.2.5, we further investigate edges cases using the Synbols dataset under different types of biases and aleatoric uncertainty.

Here, we consider both BALD and predictive entropy as scoring functions. We examine other scoring functions on Repeated-MNIST in appendix E.2.2.1 and observe similar results. For the sake of legible figures, we focus on power acquisition in this section, as it fits BALD and entropy best: the scores are non-negative, and zero scores imply uninformative samples. We show that all three methods (power, softmax, softrank) perform similarly in appendix E.3.

We are not always able to compare to BADGE and BatchBALD because of computational limitations of those methods. BatchBALD is computationally infeasible for large acquisition sizes (> 10) because of time constraints, cf. Table 5.1. When possible, we use BatchBALD with acquisition size 5 as baseline. Similarly, BADGE runs out of memory for large dataset sizes, such as EMNIST 'ByMerge' with 814,255 examples.

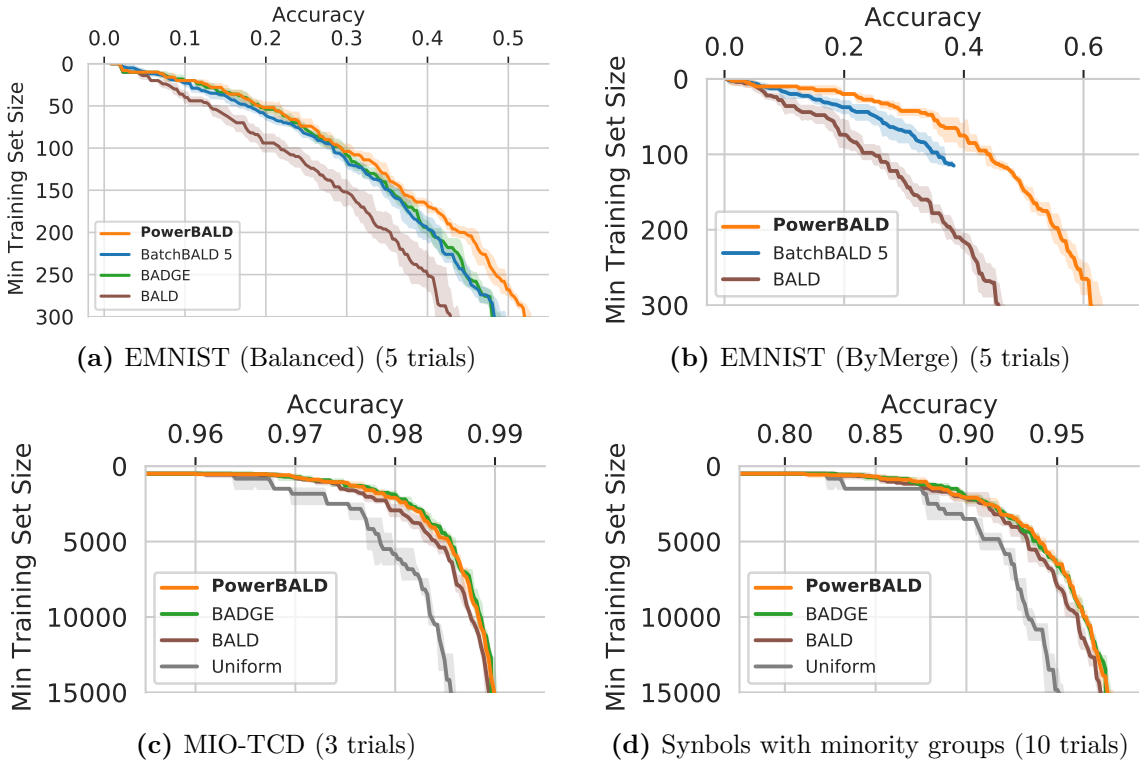
Figures interpolate linearly between available points, and we show 95% confidence intervals.

Experimental Setup & Compute. We document the experimental setup and model architectures in detail in appendix E.2.1. Our experiments used about 25,000 compute hours on Titan RTX GPUs.

Runtime Measurements. We emphasize that the stochastic acquisition strategies are much more computationally efficient compared to specialized batch-acquisition approaches like BADGE and BatchBALD. Runtimes, shown in Table 5.1, are essentially identical for top-`K` and the stochastic versions. Both are orders of magnitude faster than BADGE and BatchBALD even for small batches. Unlike those methods, stochastic acquisition scales *linearly* in pool size and *logarithmically* in acquisition size. Runtime numbers do not include the cost of retraining models (identical in each case). The runtimes for top-`K` and stochastic acquisition appear constant over `K` because the execution time is dominated by fixed-cost memory operations. The synthetic dataset used for benchmarking has 4,096 features, 10 classes, and 10,000 pool points.

Repeated-MNIST. Repeated-MNIST (§4) duplicates MNIST a specified number of times and adds Gaussian noise to prevent perfect duplicates. Redundant data are incredibly common in industrial applications but are usually removed from standard benchmark datasets. The controlled redundancies in the dataset allow us to showcase pathologies in batch acquisition methods. We use an acquisition size of 10 and 4 dataset repetitions.

Figure 5.2 shows that PowerBALD outperforms top-`K` BALD. While much cheaper computationally, cf. Table 5.1, PowerBALD also outperforms BADGE and even performs on par with BatchBALD. For BatchBALD, we use an acquisition size of 5, and for BADGE of 20. Note that BatchBALD performs better for smaller acquisition sizes while BADGE (counterintuitively) performs better for larger ones; see Figure E.2 in the appendix for an ablation. BatchBALD, BALD, and the stochastic variants all



(a) EMNIST (Balanced) (5 trials)

(b) EMNIST (ByMerge) (5 trials)

(c) MIO-TCD (3 trials)

(d) Synbols with minority groups (10 trials)

Figure 5.3: *Performance on various datasets.* BatchBALD took infeasibly long on these datasets & acquisition sizes. **((a))** *EMNIST 'Balanced':* On 132k samples, PowerBALD (acq. size 10) outperforms BatchBALD (acq. size 5) and BADGE (acq. size 40). **((b))** *EMNIST 'ByMerge':* On 814k samples, PowerBALD (acq. size 10) outperforms BatchBALD (acq. size 5). BADGE (not shown) OOM'ed, and BatchBALD took > 12 days for 115 acquisitions. **((c))** *MIO-TCD:* PowerBALD performs better than BALD and on par with BADGE (all acq. size 100). **((d))** *Synbols with minority groups:* PowerBALD performs on par with BADGE (all acq. size 100).

become equivalent for acquisition size 1 when individual points are sampled, which performs best (§4).

Computer Vision: EMNIST. EMNIST [Cohen et al., 2017] contains handwritten digits and letters and comes with several splits: we examine the 'Balanced' split with 131,600 samples in Figure 5.3(a)[2] and the 'ByMerge' split with 814,255 samples in Figure 5.3(b). Both have 47 classes. We use an acquisition size of 5 for BatchBALD, of 40 for BADGE, and of 10 otherwise.

We see that the stochastic methods outperform BatchBALD on it and both BADGE and BatchBALD on 'Balanced' (Figure 5.3(a)). They do not have any issues with the huge pool set in 'ByMerge' (Figure 5.3(b)). In the appendix, Figures E.16 and E.17 show results for all three stochastic extensions, and Figure E.8 shows an ablation of different acquisition batch sizes for BADGE. For 'ByMerge', BADGE ran out of memory on our machines, and BatchBALD took more than 12 days for 115 acquisitions when we halted execution.

[2]This result exactly reproduces BatchBALD's trajectory in Figure 7 from §4.

Computer Vision: MIO-TCD. The Miovision Traffic Camera Dataset (MIO-TCD) [Luo et al., 2018] is a vehicle classification and localization dataset with 648,959 images designed to exhibit realistic data characteristics like class imbalance, duplicate data, compression artifacts, varying resolution (between 100 and 2,000 pixels), and uninformative examples; see Figure E.1 in the appendix. As depicted in Figure 5.3(c), PowerBALD performs better than BALD and essentially matches BADGE despite being much cheaper to compute. We use an acquisition size of 100 for all methods.

Computer Vision: Synbols. Synbols [Lacoste et al., 2020] is a character dataset generator which can demonstrate the behavior of batch active learning under various edge cases [Lacoste et al., 2020; Branchaud-Charron et al., 2021]. In Figure 5.3(d), we evaluate PowerBALD on a dataset with minority character types and colors. PowerBALD outperforms BALD and matches BADGE. Further details as well as an examination of the 'spurious correlation' and 'missing symbols' edge cases [Lacoste et al., 2020; Branchaud-Charron et al., 2021] can be found in appendix E.2.5.

Natural Language Processing: CLINC-150. We perform intent classification on CLINC-150 [Larson et al., 2019], which contains 150 intent classes plus an out-of-scope class. This setting captures data seen in production for chatbots. We fine-tune a pretrained DistilBERT model from HuggingFace [Dosovitskiy et al., 2020] on CLINC-150 for 5 epochs with Adam as optimizer. In appendix E.2.6, we see that PowerEntropy shows strong performance. This demonstrates that our technique is domain independent and can be easily reused for other tasks.

Summary. We have verified that stochastic acquisition functions outperform top-`K` batch acquisition in several settings and perform on par with more complex methods such as BADGE or BatchBALD. Moreover, we refer the reader to Jesson et al. [2021], Murray et al. [2021], Tigas et al. [2022], Holmes et al. [2022] for additional works that use the proposed stochastic acquisition functions from this chapter and provide further empirical validation.

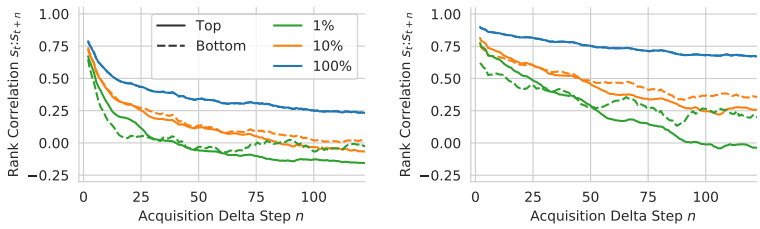
5.5 Further Investigations

In this section, we validate our assumptions about the underlying score dynamics by examining the score rank correlations across acquisitions. We further hypothesize about when top-`K` acquisition is the most detrimental to active learning.

Rank Correlations Across Acquisitions. Our method is based on assuming: (1) the acquisition scores s_t at step t are a proxy for scores $s_{t'}$ at step $t' > t$; (2) the larger $t' - t$ is, the worse a proxy s_t is for s'_t; (3) this effect is the largest for the most informative points.

We demonstrate these empirically by examining the Spearman rank correlation between scores during acquisition. Specifically, we train a model for n steps using BALD as single-point acquisition function. We compare the rank order at each step to the starting rank order at step t.

Figure 5.1 shows that acquisition scores become less correlated as more points are acquired. Figure 5.4(a) shows this in more detail for the top and bottom 1%, 10% or 100% of scorers of the test set across acquisitions starting at step $t = 0$ for a model initialized with 20 points. The top-10% scoring points (solid green) quickly become uncorrelated across acquisitions and even become *anti-correlated*. In contrast, the points overall (solid blue) correlate well over time (although they have a much weaker



(a) $t = 0$ **(b)** $t = 100$

Figure 5.4: *Rank correlations for BALD scores on MNIST between the initial scores and future scores of the top- or bottom-scoring 1%, 10% and 100% of test points (smoothed with a size-10 Parzen window).* The rank order decorrelates faster for the most informative samples and in the early stages of training. The top-1% scorers' ranks *anti-correlate* after roughly 40 (100) acquisitions unlike the bottom-1%. Later in training, the acquisition scores stay more strongly correlated. This suggests *the acquisition size could be increased later in training.*

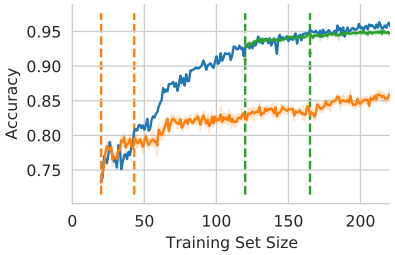


Figure 5.5: *Top-K acquisition hurts less later in training (BALD on MNIST).* At $t \in \{20, 100\}$ (blue), we keep acquiring samples using the BALD scores from those two steps. At $t = 20$ (orange), the model performs well for ≈ 20 acquisitions; at $t = 120$ (green), for ≈ 50; see §5.5.

training signal on average). This result supports all three of our hypotheses.

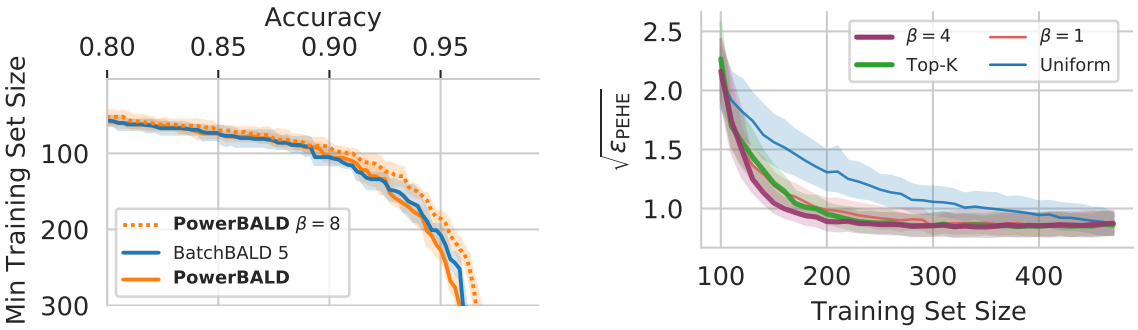
At the same time, we see that as training progresses, and we converge towards the best model, the order of scores becomes more stable across acquisitions. In Figure 5.4(b) the model begins with 120 points ($t = 100$), rather than 20 ($t = 0$). Here, the most informative points are less likely to change their rank—even the top-1% ranks do not become *anti-correlated*, only uncorrelated. Thus, we hypothesize that further in training, we might be able to choose larger `K`.

Increasing Top-K Analysis. Another way to investigate the effect of top-`K` selection is to freeze the acquisition scores during training and then continue single-point 'active learning' as if those were the correct scores. Comparing this to the performance of regular active learning with updated single-point scores allows us to examine how well earlier scores perform as proxies for later scores. We perform this toy experiment on MNIST, showing that freezing scores early on greatly harms performance while doing it later has only a small effect (Figure 5.5). For frozen scores at a training set size of 20 (73% accuracy, $t = 0$), the accuracy matches single-acquisition BALD up to a training set size of roughly 40 (dashed orange lines) before diverging to a lower level. But when freezing the scores of a more accurate model, at a training set size of 120 labels (93% accuracy, $t = 100$), selecting the next fifty points according to those frozen scores performs indistinguishably from step-by-step acquisition (dashed green lines). This result shows that top-`K` acquisition hurts less later in training but can negatively affect performance at the beginning of training.

These observations lead us to ask whether we could dynamically change the acquisition size: with smaller acquisition batches at the beginning and larger ones towards the end of active learning. We leave the exploration of this for future work.

5.5.1 Ablation: Changing β

So far, we have set $\beta = 1$ in the spirit of providing a simple baseline without additional hyperparameters. The results above show that this already works well and matches



(a) Repeated-MNISTx4 **(b)** IHDP

Figure 5.6: *Effect of changing β.* **((a))** *Repeated-MNISTx4 (5 trials):* PowerBALD outperforms BatchBALD for $\beta = 8$. **((b))** *IHDP (400 trials):* At high temperature ($\beta = 0.1$), CausalBALD with power acquisition is like random acquisition. As the temperature decreases, the performance improves (lower $\sqrt{\epsilon_{\text{PEHE}}}$), surpassing top-K acquisition.

the performance of much more expensive methods, raising questions about their value. In addition, however, tuning β may be able to further improve performance. Next, we show that other values of β can yield even higher performance on Repeated-MNIST and when estimating causal treatment effects; we provide additional results in appendix E.4.

Repeated-MNIST. In Figure 5.6(a), we see that for PowerBALD the best-performing value, $\beta = 8$, outperforms BatchBALD.

Causal Treatment Effects: Infant Health Development Programme. Active learning for Conditional Average Treatment Effect (CATE) estimation Heckman et al. [1997, 1998]; Hahn [1998]; Abrevaya et al. [2015] on data from the Infant Health and Development Program (IHDP) estimates the causal effect of treatments on an infant's health from observational data. Statistical estimands of the CATE are obtainable from observational data under certain assumptions. Jesson et al. [2021] show how to use active learning to acquire data for label-efficient estimation. Among other subtleties, this prioritizes the data for which matched treated/untreated pairs are available.

We follow the experiments of Jesson et al. [2021] on both synthetic data and the semisynthetic IHDP dataset [Hill, 2011], a commonly used benchmark for causal effects estimation. In Figure 5.6(b) we show that power acquisition performs significantly better than both top-K and uniform acquisition, using an acquisition size of 10 in all cases with further. We provide additional results on semisynthetic data in appendix E.4.2. Note that methods such as BADGE and BatchBALD are not well-defined for causal-effect estimation, while our approach remains applicable and is effective when fine-tuning β: BatchBALD and BADGE are specifically designed for active learning given (classification) predictions, which is not the same as estimating causal effects.

Performance on these tasks is measured using the expected *Precision in Estimation of Heterogeneous Effect (PEHE)* [Hill, 2011] such that $\sqrt{\epsilon_{\text{PEHE}}} = \sqrt{\mathbb{E}[(\tilde{\tau}(\mathbf{X}) - \tau(\mathbf{X}))^2]}$ [Shalit et al., 2017] where $\tilde{\tau}$ is the estimated CATE and τ is CATE (i.e. a form of RMSE).

Limitations. Although we highlight the possibility for future work to adapt β to specific datasets or score functions, our aim is not to offer a practical recipe for this to practitioners. Our focus is on showing how even the simplest form of stochastic acquisition already raises questions for some recent more complex methods.

5.6 Discussion

We have demonstrated a surprisingly effective and efficient baseline for batch acquisition in active learning. The presented stochastic extensions are orders of magnitude faster than sophisticated batch-acquisition strategies like BADGE and BatchBALD while retaining comparable performance in many settings. Compared to the flawed top-`K` batch acquisition heuristic, it is never worse: we see no reason to continue using top-`K` acquisition.

Importantly, this chapter raises serious questions about these current methods. If they fail to outperform such a simple baseline in a wide range of settings, do they model the interaction between points sufficiently well? If so, are the scores themselves unreliable?

At the same time, this presented framework opens doors for improved methods. Although our stochastic model is put forward for its computational and mathematical simplicity, future work could explore more sophisticated modelling of the predicted score changes that take the current model and dataset into account. In its simplest form, this might mean adapting the temperature of the acquisition distribution to the dataset or estimating it online. Our experiments also highlight that the acquisition size could be dynamic, with larger batch sizes acceptable later in training.

The whole universe is contained in a single flower.

Toshiro Kawase

6

Marginal and Joint Cross-Entropies & Predictives

Beyond deep ensembles [Lakshminarayanan et al., 2017], more principled methods of deep learning that attempt to be (approximately) Bayesian, commonly referred to as (approximate) *Bayesian Neural Networks (BNN)*, have arguably not lived up to their full potential [Ovadia et al., 2019; Beluch et al., 2018]. This might be because the focus in their evaluation has been on marginal predictions $\mathrm{q}(y \mid \boldsymbol{x})$, where they can only provide *marginal*$^{\text{D}}$ improvements over unprincipled regular NNs.

Yet the strength of a Bayesian approach for deep learning might not solely lie in marginal predictions but in joint predictions and in allowing for online learning via online Bayesian inference. *Online Bayesian inference (OBI)* refers to incorporating additional data into the posterior predictive *without* explicitly retraining in the common sense, i.e. by computing gradients and optimizing the model parameters further[1]. This could offer important performance benefits for applications that would otherwise require repeated retraining like active learning and could have important implications for how we could use large supervised models in production: currently, they are seen as strictly static; however, online Bayesian inference would allow them to dynamically adapt to new data on the fly.

Generally, the difference between an approximate BNN and a regular NN is that the former assumes a distribution $\mathrm{q}(\omega)$ over the model parameters ω, where $\mathrm{q}(\omega)$ approximates the Bayesian posterior $\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$, which is the optimal distribution given prior information $\mathrm{p}(\omega)$ and training data $\mathcal{D}^{\text{train}}$: $\mathrm{q}(\omega) \approx \mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$.

Online Bayesian Inference. To incorporate new data $\{y_i, \boldsymbol{x}_i\} \sim \hat{\mathrm{p}}_{\text{true}}(y, \boldsymbol{x})^n$, via online Bayesian inference, we simply apply Bayes' theorem: for a test point $\boldsymbol{x}$, the predictive $\mathrm{q}(y \mid \boldsymbol{x}, y_n, \boldsymbol{x}_n, \dots, y_1, \boldsymbol{x}_1)$ is proportional to its joint predictive. We obtain:

$$\mathrm{q}(y \mid x, y_n, x_n, \dots, y_1, x_1) = \frac{\mathrm{q}(y, y_n, \dots, y_1 \mid x, x_n, \dots, x_1)}{\mathrm{q}(y_n \dots, y_1 \mid x_n, \dots, x_1)} \tag{6.1}$$

$$\propto \mathrm{q}(y, y_n, \dots, y_1 \mid x, x_n, \dots, x_1). \tag{6.2}$$

We can thus use a joint predictive $\mathrm{q}(y, y_1, \dots, y_n \,|\, \boldsymbol{x}, \boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ to incorporate fixed$\{y_i, \boldsymbol{x}_i\}^n$ and make predictions for $\boldsymbol{x}$ without explicit retraining.

[1] Maddox et al. [2021] have referred to this approach as 'Online Variational Conditioning' in the context of Gaussian processes. However, we would argue that in the context of this chapter OBI is the better term because there is nothing variational about it. Indeed, OBI applies Bayesian inference to an 'imperfect' variational intermediate posterior instead of performing Bayesian inference end-to-end from the original prior.

Hence, for online Bayesian inference, we require joint predictives, which only Bayesian methods can give us:[2] through BNNs in the parametric case or through Gaussian processes in the non-parameteric case, for example. This strongly contrasts with marginal predictives which can also be modelled by regular NNs.

Can we perform online Bayesian inference well for high-dimensional inputs and parameters using current approximate BNNs? The quality of the resulting predictions crucially depends on the joint predictives. However, computing joint predictives can be challenging. For example, in Bayesian literature, the joint predictive of all samples in the training set marginalized over the prior distribution is just the well-known marginal likelihood, which can be used for model selection [MacKay, 2003; Lyle et al., 2020; Llorente et al., 2023], and is known to be difficult to estimate in high-dimensional parameter spaces [Lotfi et al., 2022]. Similar challenges can be expected for performing online Bayesian inference using approximate posterior distributions.

Based on recent works by [Wen et al., 2021; Osband et al., 2022b], we examine the joint predictives for online Bayesian inference using naive prior sampling. However, we find negative results when using MC dropout [Gal and Ghahramani, 2016b].

Relevant Literature. Osband et al. [2022b]; Wen et al. [2021]; Osband et al. [2021b,a] mention the importance of joint predictives in the context of combinatorial decision problems, sequential predictions and multi-armed bandits in low dimensions. Compared to these previous works, we explore important connections to supervised learning, e.g. in active learning [Atlas et al., 1989; Settles, 2010] and Bayesian optimal experimental design [Lindley, 1956; Foster, 2022], and focus on online Bayesian inference in high dimensions using deep neural networks. This is also an important difference to Maddox et al. [2021] which examines online Bayesian inference in the context of Gaussian processes.

In addition to these works, we clarify that both marginal and joint cross-entropies have their use, and it is not the case that one is always preferable over the other. Simply put, we argue that they capture different quantities that are separately useful in offline and online learning, and we combine them to evaluate the performance of online Bayesian inference using approximate BNNs. We provide more details in §6.5.

Marginal Cross-Entropy. As will become evident, the marginal cross-entropy for a fixed predictive model captures the expected performance (*under log loss*) when the model does not adapt to data at test time. For supervised tasks, the marginal cross-entropy is what is commonly referred to as cross-entropy loss and represents common practice: we obtain a fixed set of parameters by training the model on a training set, and we re-use these parameters at test time without any further updates. The performance of the model does not change as it observes more test data, and there is no feedback loop of any sort. In this *offline learning* setting, the marginal cross-entropy is the right choice to estimate performance.

Joint Cross-Entropy. On the other hand, the joint cross-entropy for a predictive model captures the performance in a *sequential learning* setting, where sequential model updates take place. Here, the parameter distribution $q(\omega)$ serves as a prior for online Bayesian inference. This fits the context used in Wen et al. [2021] in which the model makes a prediction for the next step, observes the outcome, and

[2]For consistent joint predictives, adhering to the chain rule of probability, a model needs to adhere to the "Bayesian update rule", i.e. Bayes' theorem, and thus is Bayesian. See also Equation 6.7 below.

then updates the model (agent).

Applications & Experiments. We will also see that the joint predictive is important for data selection in active learning and active sampling. This chapter also connects several recent works [Wen et al., 2021; Osband et al., 2022b] with active learning and active sampling and present new realistic and challenging experimental settings. Most importantly, we examine online Bayesian inference within these contexts as it allows us to avoid *retraining across acquisitions*.

Notation. This chapter mentions many cross-entropies. We will use a more concise notation to save space:

$$\mathrm{H}_{\mathrm{p}\,\|\,\mathrm{q}}[\cdot] \triangleq \mathrm{H}(\mathrm{p}(\,\cdot\,) \,\|\, \mathrm{q}(\,\cdot\,)) = \mathbb{E}_{\mathrm{p}(\cdot)}[-\log \mathrm{q}(\cdot)]. \tag{6.3}$$

For example: $\mathrm{H}_{\mathrm{p}\,\|\,\mathrm{q}}[Y \mid x] = \mathbb{E}_{\mathrm{p}(y|x)}[-\log \mathrm{q}(y \mid x)]$.

Background. The setting in this chapter deviates slightly from the one introduced in §1.2.2: *The important difference in this chapter to §1.2.2 is that we investigate how* q *compares to* p *and do not ignore the difference.* That is, we assume an underlying parametric predictive model $\mathrm{p}(y \mid \boldsymbol{x}, \omega)$ for input samples $\boldsymbol{x}$ with targets or labels y with a prior parameter distribution $\mathrm{p}(\omega)$ over ω, i.e. our model is Bayesian. As noted previously, capturing the true posterior distribution $\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$ is infeasible, and we assume we have an approximate distribution $\mathrm{q}(\omega) \approx \mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$ that we use instead of the true posterior. For example, $\mathrm{q}(\omega)$ could be based on a deep ensemble [Lakshminarayanan et al., 2017] as a mixture of Dirac delta distributions positioned at the parameters of individually trained ensemble members, or it could be an MC dropout model that is trained using variational inference [Gal and Ghahramani, 2016a]. We use $\mathrm{q}(y \mid \boldsymbol{x})$ to denote the predictions after marginalizing over $\mathrm{q}(\omega)$: $\mathrm{q}(y \mid \boldsymbol{x}) = \mathbb{E}_{\mathrm{q}(\omega)}[\mathrm{p}(y \mid \boldsymbol{x}, \omega)]$. Note that the underlying discriminative model (or the likelihood function) $\mathrm{p}(y \mid \boldsymbol{x}, \omega)$ remains the same—we only exchange the distribution over its parameters ω.

6.1 Marginal and Joint Cross-Entropy

We begin by contrasting marginal and joint predictive cross-entropies and revisiting how they are useful for offline and online learning separately.

Marginal Cross-Entropy. Given an underlying, possibly empirical, data distribution $\hat{\mathrm{p}}_{\text{train}}(\boldsymbol{x}, y)$, the *marginal cross-entropy* is:

$$\begin{aligned} \mathrm{H}_{\hat{\mathrm{p}}_{\text{train}}\,\|\,\mathrm{q}}[Y \mid X] &= \mathbb{E}_{\hat{\mathrm{p}}_{\text{train}}(x,y)}[-\log \mathbb{E}_{\mathrm{q}(\omega)}[\mathrm{p}(y \mid x, \omega)]] \\ &= \mathbb{E}_{\hat{\mathrm{p}}_{\text{train}}(x,y)}[-\log \mathrm{q}(y \mid x)], \end{aligned} \tag{6.4}$$

where we use $\mathrm{H}_{\hat{\mathrm{p}}_{\text{train}}\,\|\,\mathrm{q}}[Y \mid X]$ to denote the cross-entropy. This cross-entropy is the population loss when $\mathrm{q}(\omega)$ is not updated after seeing new samples. Each sample $\boldsymbol{x}, y$ is treated independently. Hence, the marginal cross-entropy captures the expected performance in an *offline learning* setting.

Joint Cross-Entropy. On the other hand, given an initial parameter distribution $\mathrm{q}(\omega)$ above, the joint cross-entropy measures how well the parameter distribution can *adapt* to new data $\mathcal{D}$.

To show how this connects to joint cross-entropies, we can look at the joint cross-entropy of specific samples $\mathcal{D} = \{y_i, \boldsymbol{x}_i\}_{i=1}^n$ (without taking an expectation).

The joint cross-entropy for these specific samples is just the sum of (negative) log marginal likelihoods using the chain rule, where each y_i is conditioned on all 'previous' observations $\mathcal{D}_{<i}$:

$$\begin{aligned}
\mathrm{H}_{\hat{\mathrm{p}}_{\text{train}} \| \mathrm{q}}[y_1, \ldots, y_n \mid x_1, \ldots, x_n] &= -\log \mathrm{q}(y_1, \ldots, y_n \mid x_1, \ldots, x_n) && (6.5)\\
&= -\log \prod_i \mathrm{q}(y_i \mid x_i, y_{i-1}, x_{i-1}, \ldots, y_1, x_1) && (6.6)\\
&= -\sum_i \log \mathrm{q}(y_i \mid x_i, \mathcal{D}_{<i}) && (6.7)\\
&= \sum_i \mathrm{H}_{\hat{\mathrm{p}}_{\text{train}} \| \mathrm{q}}[y_i \mid x_i, \mathcal{D}_{<i}], && (6.8)
\end{aligned}$$

where $\mathcal{D}_{<i}$ denotes "$y_{i-1}, \boldsymbol{x}_{i-1}, \ldots, y_1, \boldsymbol{x}_1$", and the marginal predictive is: $\mathrm{q}(y_i \mid x_i, \mathcal{D}_{<i}) = \mathbb{E}_{\mathrm{q}(\omega \mid \mathcal{D}_{<i})}[\mathrm{p}(y_i \mid x_i, \omega)]$. Semantically, we compute the following in each iteration of the sum: we *update* the parameter posterior $\mathrm{q}(\omega \mid \mathcal{D}_{<i})$, compute the losses for our predictions at outcomes y_i for $\boldsymbol{x}_i$, and then include $y_i, \boldsymbol{x}_i$ in our observed data. We will denote this as the *online learning setting*.

When we are interested in the expected loss given arbitrary data, we can compute the following joint cross-entropy:

$$\begin{aligned}
OLL(n) &\triangleq \mathbb{E}_{(x_i, y_i)_{i=1}^n \sim \hat{\mathrm{p}}_{\text{true}}(x_i, y_i)^n}[-\log \mathrm{q}(y_1, \ldots, y_n \mid x_1, \ldots x_n)] && (6.9)\\
&= \mathbb{E}_{(x_i, y_i)_{i=1}^n \sim \hat{\mathrm{p}}_{\text{true}}(x_i, y_i)^n}[\mathrm{H}_{\hat{\mathrm{p}}_{\text{train}} \| \mathrm{q}}[y_1, \ldots, y_n \mid x_1, \ldots x_n]] && (6.10)\\
&= \mathrm{H}_{\hat{\mathrm{p}}_{\text{train}} \| \mathrm{q}}[Y_1, \ldots, Y_n \mid X_1, \ldots X_n], && (6.11)
\end{aligned}$$

where OLL stands for "online learning loss".

Connection to the Conditional Cross-Entropy Rate. As an aside, if we let $n \to \infty$, we also have $OLL(n) \to \infty$. This is not helpful, so instead we can look at the average: $\frac{1}{n} OLL(n)$. In the limit, this average is just the *cross-entropy rate*:

$$\mathrm{H}_{\hat{\mathrm{p}}_{\text{train}} \| \mathrm{q}}[\mathcal{Y} \mid \mathcal{X}] \triangleq \lim_{n \to \infty} \frac{1}{n} \mathrm{H}_{\hat{\mathrm{p}}_{\text{train}} \| \mathrm{q}}[Y_1, \ldots, Y_n \mid X_1, \ldots X_n], \tag{6.12}$$

which we define analogously to the entropy rate in Cover and Thomas [2005].[3]

Summary. The marginal cross-entropy is useful for offline learning as it predicts the performance of a fixed model on the data distribution. The joint cross-entropy is useful for online learning as it predicts the performance of a model as it adapts to additional data. Hence, it is important for sequential decision-making.

6.2 Online Bayesian Inference

To see what we mean by incorporating new data, assume we have sampled n additional points $\boldsymbol{x}_i, y_i \sim \hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}_i, y_i)$. Traditionally, we would now update the posterior approximation $\mathrm{q}(w)$ to take this new data into account for our predictions at future test points. However, this can be prohibitively expensive—especially in applications that require frequent retraining. Instead, online Bayesian inference allows Bayesian models to adapt their predictions without explicitly updating the posterior approximation.

[3] Cf. the entropy rate, which is: $\mathrm{H}[\mathcal{X}] = \lim_{n \to \infty} \frac{1}{n} \mathrm{H}[X_1, \ldots X_n]$.

Following Equation 6.1, for a test point $\boldsymbol{x}$, the predictive $\mathrm{q}(y \mid \boldsymbol{x}, y_n, \boldsymbol{x}_n, \ldots, y_1, \boldsymbol{x}_1)$ is proportional to the joint predictive:

$$\mathrm{q}(y \mid x, y_n, x_n, \ldots, y_1, x_1) = \frac{\mathrm{q}(y, y_n, \ldots, y_1 \mid x, x_n, \ldots, x_1)}{\mathrm{q}(y_n \ldots, y_1 \mid x_n, \ldots, x_1)} \tag{6.13}$$

$$\propto \mathrm{q}(y, y_n, \ldots, y_1 \mid x, x_n, \ldots, x_1), \tag{6.14}$$

since the normalization constant $\mathrm{q}(y_n \ldots, y_1 \mid \boldsymbol{x}_n, \ldots, \boldsymbol{x}_1)$ is independent of y and $\boldsymbol{x}$. Hence, this allows us to make predictions that take into account new data *without explicit retraining* by simply computing the joint predictive of the test point and newly observed data.

We refer to this as *online Bayesian inference (OBI)*. While this inference is precisely Bayesian, $\mathrm{q}(\omega)$ is commonly only an approximate posterior, and thus the quality of this inference depends on the properties of the approximation and how we estimate the joint predictive.

The simplest approach to estimate the joint predictive is via sampling, which applies to e.g. Monte-Carlo dropout, deep ensembles, and deep ensembles with prior functions [Gal and Ghahramani, 2016a; Lakshminarayanan et al., 2017; Osband et al., 2018], by factorizing the joint:

$$\mathrm{q}(y, y_n, \ldots, y_1 \mid x, x_n, \ldots, x_1) = \mathbb{E}_{\mathrm{q}(\omega)}[\mathrm{p}(y, y_n, \ldots, y_1 \mid x, x_n, \ldots, x_1, \omega)] \tag{6.15}$$

$$= \mathbb{E}_{\mathrm{q}(\omega)}\left[\mathrm{p}(y \mid x, \omega) \prod_{i=1}^{n} \mathrm{p}(y_i \mid x_i, \omega)\right]. \tag{6.16}$$

Thus, if we draw fixed parameter samples $\omega_j \sim \mathrm{q}(\omega)$, we can pre-compute $\prod_{i=1}^{n} \mathrm{p}(y_i \mid \boldsymbol{x}_i, \omega_j)$ for each j and estimate the joint predictive.

Finally, we can view $\prod_{i=1}^{n} \mathrm{p}(y_i \mid \boldsymbol{x}_i, \omega)$ as unnormalized importance weights:

$$\mathrm{q}(\omega) \prod_{i=1}^{n} \mathrm{p}(y_i \mid x_i, \omega) \propto \mathrm{q}(\omega \mid y_n, x_n, \ldots, y_1, x_1), \tag{6.17}$$

and hence, overall:

$$\mathbb{E}_{\mathrm{q}(\omega)}\left[\mathrm{p}(y \mid x, \omega) \prod_{i=1}^{n} \mathrm{p}(y_i \mid x_i, \omega)\right] \propto \mathbb{E}_{\mathrm{q}(\omega \mid y_n, x_n, \ldots, y_1, x_1)}[\mathrm{p}(y \mid x, \omega)]. \tag{6.18}$$

To evaluate the performance, we can use these predictions to compute the following marginal cross-entropy which incorporates the additional samples using OBI:

$$\mathrm{H}_{\hat{\mathrm{p}}_{\mathrm{train}} \| \, \mathrm{q}}[Y \mid X, y_n, x_n, \ldots, y_1, x_1], \tag{6.19}$$

where X, Y are sampled from the data distribution. Comparing this entropy with the performance of a fully retrained model allows us to obtain a practical estimate for the quality of the approximate posterior $\mathrm{q}(\omega)$.

6.3 New Evaluations & Applications

We suggest new experimental settings that allow us to evaluate the quality of the joint predictive and compare the settings to ones suggested in prior work.

6.3.1 Performance in Active Learning and Active Sampling Methods

A conceptually simple set of downstream tasks is to evaluate the performance of the joint predictives in different active learning or active sampling settings using different approximate model architectures (e.g. based on Epistemic Neural Networks Osband et al. [2021a] as abstraction). Wang et al. [2021] show that performance for transductive active learning is correlated to the quality of the joint predictive. We recall that we noticed the same in §4 for batch active learning with non-transductive acquisition functions, i.e. BatchBALD's performance heavily on the number of MC dropout samples which informs the quality of the joint predictive.

We suggest, similar to Repeated-MNIST (§4), to duplicate the underlying pool sets or to simply *allow the same sample to be selected multiple times*. Importantly, approximate BNNs that do not provide good joint predictives will greedily select the same sample over and over again or degrade to uninformed data acquisitions. This avoids an issue pointed out by Wang et al. [2021] and Osband et al. [2022b] as we explain next.

Connection to Total Correlation. Wang et al. [2021] argue that the joint cross-entropy is dominated by the sum of the individual marginal cross-entropy scores. This is equivalent to saying that the total correlation between samples is negligible since the difference between the joint cross-entropy and its individual marginal cross-entropies for specific $y_i, \boldsymbol{x}_i$ is just the *total correlation*:

$$\mathrm{TC}_\mathrm{q}[y_1, \ldots, y_n \mid x_1, \ldots, x_n] \triangleq \sum_i \mathrm{H}_\mathrm{q}[y_i \mid x_i] - \mathrm{H}_\mathrm{q}[y_1, \ldots, y_n \mid x_1, \ldots, x_n]. \tag{6.20}$$

The total correlation measures the amount of information shared between the samples.

For random batches, the total correlation is, indeed, likely going to be negligible because most random batches are not very informative overall, and importantly, on curated datasets, they are most likely uncorrelated as it is unlikely that observing y_i, $\boldsymbol{x}_i$ in the batch informs prediction for y_j, $\boldsymbol{x}_j$ for most $j \neq i$ as curated datasets are usually as diverse as possible.

Only with increasing redundancy in the dataset, e.g. by duplicating samples like in Repeated-MNIST (§4), random batches will become more correlated on average and the total correlation larger.

This setup is similar to the dyadic sampling proposed by Osband et al. [2022b] which repeatedly samples y_i^j for $\boldsymbol{x}_i$, with $i \in \{1, 2\}$ and evaluates the joint predictive. However, this setting in essence only measures the ability of the approximate model to perform Bayesian updates on two fixed training samples at a time. Hence, we suggest that a better adaption to evaluate joint predictives using active learning is to duplicate the dataset or to simply allow the same sample to be selected multiple times.

6.3.2 Performance of Online Bayesian Inference

As a practical "ground truth", we can compare the performance of retrained models after acquiring additional samples with the performance of OBI as explained in §6.2.

A particular challenging scenario for OBI is to use acquisition sequences $(\boldsymbol{x}_i, y_i)_{i=1}^T$ that were collected using active learning or active sampling on the dataset. We evaluate OBI on models trained at different $\mathcal{D}_t^{\text{train}} = \{y_i, \boldsymbol{x}_i\}_{i=1}^t$ with increasing subsets of online data $\{y_i, \boldsymbol{x}_i\}_{i=t+1}^T$ from these acquisition sequences. This scenario is particularly

challenging for OBI because the sequence of acquisition is selected to result in large changes in the predictives and posterior distributions.

The average performance difference between OBI and fully retrained models across different training acquisition sequences will tell us how good a given joint predictive is for "meaningful" online learning. The expectation is that for most approximate BNNs, OBI will quickly suffer from degraded performance compared to the retrained models.

That is, we compare the performance of OBI as we acquire new samples $\boldsymbol{x}_i, y_i$ to an approximate BNN retrained with the same additional data for increasing n:

$$\mathrm{H}_{\hat{\mathrm{p}}_{\text{train}} \| \, \mathrm{q}}[Y \mid X, y_n, x_n, \ldots, y_1, x_1] - \mathrm{H}_{\hat{\mathrm{p}}_{\text{train}} \| \, \mathrm{q}'}[Y \mid X], \tag{6.21}$$

where $\mathrm{q}'(\omega) \approx \mathrm{p}(\omega \mid y_n, x_n, \ldots, y_1, x_1, \mathcal{D}^{\text{train}})$ is the parameter distribution of an (approximate) BNN after retraining with the additional $y_n, \boldsymbol{x}_n, \ldots, y_1, \boldsymbol{x}_1$.

Ideally, we would compare to the predictions from the correct updated posterior distribution; however, this is infeasible in most practical scenarios. Instead, when we use an approximate BNN $\mathrm{q}'(\omega)$ that is similar to the one used for $\mathrm{q}(\omega)$, we can measure the practical degradation between OBI and retraining. Here, the ideal would be for OBI to behave exactly like a fully retrained model—even if the latter does not match exact Bayesian inference—as such an approach would be *self-consistent*. Note that with exact Bayesian inference, we would have $\mathrm{q}(\omega) = \mathrm{q}'(\omega)$, and the above would be zero.

Comparison to Dyadic Sampling. Unlike Osband et al. [2022b] which focuses on selecting labels for dyadic samples repeatedly, this experiment setting is both more practical and more insightful: active learning picks samples that are the most informative and are supposed to update the posterior the most. This is because, for informative samples, we would expect the changes in model predictions to be the largest. Hence, one could expect that these samples pose the most significant challenge to approximate BNNs and their joint predictives. Ideally, we would hope that OBI would keep up with retrained model, but this might prove to be challenging in high-dimensional scenarios.

Sample Selection Bias. The suggested evaluation is orthogonal to any sample selection bias that is added through the data acquisition process itself as we use the same training data at each step for both OBI and retraining in Equation 6.21. Specifically, Farquhar et al. [2021] observed that active learning introduces a bias by sampling from the data distribution using an acquisition function and not uniformly.

6.3.3 Application: Active Learning with Online Bayesian Inference

In many settings, retraining models for when only few new samples are added is prohibitively expensive. This motivates batch active learning, where batches are acquired instead of individual samples. Expanding on this, when equipped with models that perform well under OBI, one could avoid retraining models when acquiring new data by using OBI. Only when OBI degrades, fully retraining will become necessary. We can evaluate this both for individual acquisition and for batch acquisition.

6.4 Empirical Validation

Following the newly suggested experimental settings, we want to run comparisons using different kinds of approximate BNNs in active learning and active sampling settings.

Figure 6.1: *Comparison between online Bayesian inference and retraining for 5 additional samples on MNIST.* We compare the dynamics between using a random acquisition sequence and using an *active* acquisition sequence, drawn using active sampling. While OBI does not appear to be significantly worse or better on the random acquisition sequence, it markedly deteriorates when using the active acquisition sequence. OBI struggles with the informative samples that change the posterior a lot: the active acquisition sequence reaches 90% accuracy with just 70 samples, unlike the random one.

Table 6.1: *Comparison between online Bayesian inference and retraining for 5 additional samples on MNIST.* OBI performs worse than the baseline (i.e. not taking into account new samples at all) when using an active acquisition sequence. On the random acquisition sequence, it only performs just as well as not updating at all. (Mean for 5 model trials and 5 OBI trials for each.)

Baseline **vs**	OBI: +5 samples		Retraining: +5 samples	
Avg	Δ Cross-Entropy (↓)	Δ Accuracy (↑)	Δ Cross-Entropy (↓)	Δ Accuracy (↑)
Active Acquisition Sequence	0.16	-5.7%	-0.062	2.0%
Random Acquisition Sequence	0.00	0.0%	-0.075	1.8%

Moreover, given fixed acquisition sequences, determined using active learning and active sampling, we want to evaluate the difference between OBI and fully retrained models.

An initial experiment shows that OBI in high dimensions might not be feasible using simple Monte Carlo approximations of the expectations in parameter space. This is likely because of the much higher dimensionality of the problems we consider—especially in comparison to Osband et al. [2021b].

Specifically, we use an acquisition sequence created using active sampling which achieves 90% accuracy on MNIST with just 70 samples and use the same model and training setup as Kirsch and Gal [2021][4]. After every new data point (starting from 20), we evaluate a retrained model using OBI with 5 additional data points and compare it to a fully retrained model with the same 5 additional data points as well as to a model that is not retrained at all. We train the models (5 trials) for each training set size and (bootstrap) sample 10000 MC dropout samples for OBI 5 times out of 20000 MC dropout samples using consistent MC dropout (§4) for each model (5 sub-trials) to reduce the variance.

Ideally, when using OBI, we should recover the same performance as if we fully retrained the models using the additional data. However, as is visible in Figure 6.1, this is not the case when using the challenging acquisition sequence from active sampling.

Table 6.1 shows the average performance difference when using OBI with additional samples from the acquisition sequence and when fully retraining. In all cases, OBI performs worse than retraining. On the random acquisition sequence, it performs only as well as not updating at all, while on the active acquisition sequence, it always performs worse.

6.5 Related Work

The most relevant works and indeed one of the inspirations for this work are Wen et al. [2021] and Osband et al. [2022b], which are recommended reading. We see this chapter as a contribution that provides a different and differentiated position on the benefits of marginal versus joint predictives and respective cross-entropies as performance metrics and that puts greater focus on OBI.

Moreover, our suggested experimental settings expand on these prior works and draw attention to active learning and active sampling as more realistic use-cases. Measuring the error between OBI and retrained models expands on the experiments in Wen et al. [2021] while evaluating performance in active learning and active sampling on highly redundant datasets (allowing to re-select previously selected points) expands on the idea of dyadic sampling from Osband et al. [2022b].

Lastly, Wen et al. [2021] focus on the KL divergence between the exact Bayesian joint predictive and the joint predictive of an approximate Bayesian model for different numbers of samples in the joint. Our suggested experimental settings focus on evaluating downstream tasks.

Wang et al. [2021] also examine the quality of joint predictives on low-dimensional datasets using a more synthetic evaluation method, the cross-normalized log-likelihood. They also evaluate the quality of joint predictives in regression settings using active learning experiments. Our evaluation settings from §6.3 extend these.

6.6 Discussion

In this short chapter, we have revisited the difference between marginal and joint cross-entropies and predictives, clarifying in which contexts either is appropriate: for offline learning, the marginal cross-entropy is the right choice to evaluate performance while for online learning, it is the joint cross-entropy. We have shown how the joint

[4]The preprint version contains information gain experiments.

predictive plays an important role in information-theoretic acquisition functions in active learning and active sampling.

Importantly, we argue that online Bayesian inference could provide many benefits and have proposed new more practical and challenging experimental settings which expand on prior art by using active learning and active sampling.

Given the results of the presented experiment, it is an open question how much better other sampling-based approaches can be when using high-dimensional parameter spaces. Especially deep ensembles which usually provide a much smaller "sample count" (i.e. number of ensemble members) might not perform well under online Bayesian inference because the hypothesis space will be exhausted faster—even when the ensemble members are diverse. Future research will hopefully offer further experimental evaluation following §6.3, e.g. improving the quality of online Bayesian inference by studying higher quality posterior distributions such as those from HMC or efficient low-dimensional posterior approximations that might make parameter-space integrals tractable. Prior research into failures of Bayesian model averaging under model misspecification might provide further insights and paths to improvements [Minka, 2002].

Using the suggested experimental settings from this chapter (or rather the original preprint), Herde et al. [2022] suggest using last-layer Laplace. Similarly, Osband et al. [2022a] explore active learning for evaluating their proposed EpiNets with great results.

Strategy without tactics is the slowest route to victory. Tactics without strategy is the noise before defeat.

Sun Tzu, The Art of War

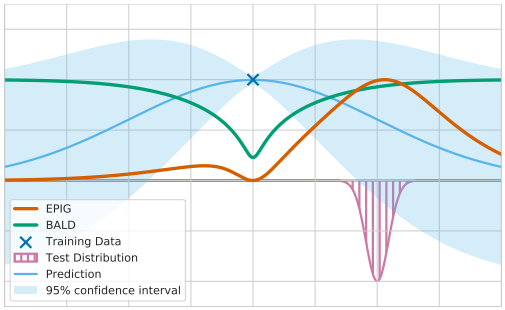
7

Prediction- & Distribution-Aware Bayesian Active Learning

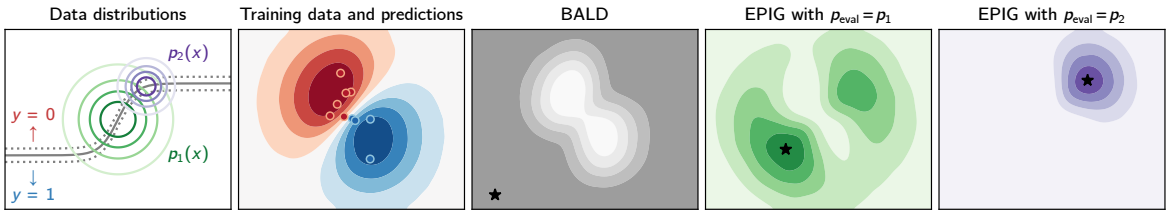
Historically the literature in active learning and Bayesian experimental design has focused on trying to maximize the expected information gain (EIG) in the model parameters. This yields the acquisition function typically known as BALD, having been popularized by a method called Bayesian active learning by disagreement [Houlsby et al., 2011], which has been successfully applied in settings including computer vision [Gal et al., 2017] and natural-language processing [Shen et al., 2018] as we have seen already in this thesis.

In this chapter, we highlight that BALD can be misaligned with our typical overarching goal of making accurate predictions on unseen inputs. In machine learning, we often care about predictions more than the parameter distribution—the parameters $\boldsymbol{\Omega}$ are not actually what we care about: they are merely a stepping stone to the predictions we will later make, and these future predictions are the ultimate quantity of interest. Why can BALD be misaligned with this, and why is this distinction important? Unfortunately, BALD neglects a crucial fact: not all information about the model parameters is equally useful when it comes to making predictions. This distinction can be surprisingly important: accurate predictions may not correspond to the most precise knowledge of the model parameters themselves. With a non-parametric model, for instance, we can gain an infinite amount of information about the model parameters without any of it being relevant to prediction on inputs of interest. More generally, different information about the model parameters may not be equal in regard to how it enables effective prediction for a particular target input distribution over possible inputs $\boldsymbol{x}$. The models we deploy are often of limited capacity or misspecified, and there are trade-offs in the performance of the model depending on what we value [Cobb et al., 2018]. In short, BALD lacks a notion of how the model will be used and so fails to ensure that the data acquired is relevant to our particular predictive task. As a result, BALD is liable to acquire labels that are informative with respect to the model parameters but not with respect to the predictions of interest.

Relevance. This has considerable practical implications. Real-world datasets are often messy, with inputs that vary widely in their relevance to a given task. Large pools of audio, images and text commonly fit this description [Ardila et al., 2020; Gemmeke et al., 2017; Mahajan et al., 2018; Radford et al., 2021; Raffel et al., 2020; Sun et al., 2017]. We may have data from different sources of varying fidelity and relevance to our task. Indeed, if we have a large unlabeled dataset generated by scraping the internet, for example, we might have a significant variance in how closely



(a) 1D Toy Example



(b) 2D Toy Example

Figure 7.1: *The expected predictive information gain (EPIG) can differ dramatically from the expected information gain in the model parameters (BALD).* **(a):** We consider a simple 1D regression task with a Gaussian process model. Larger values of the acquisition function indicate a preference for those points. While BALD is predominantly concerned with labeling inputs far away from the data it has already seen, EPIG takes into account the target input distribution (the test distribution). **(b):** BALD increases (darker shading) as we move away from the existing data, yielding a distant acquisition (star) when maximized. It seeks a global reduction in parameter uncertainty, regardless of any input distribution. In contrast, EPIG is maximized only in regions of relatively high density under the target input distribution, $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$. It seeks a reduction in parameter uncertainty only insofar as it reduces predictive uncertainty on samples from $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$. See §7.3.4 for details.

related individual data points are to the task we actually care about—if the task is to detect hate speech in text, social-media posts are much more useful than articles about set theory. Likewise, consider the outputs of large-scale experiments: if we want to predict the behavior of plasma in a new fusion-reactor configuration, results from similar configurations will likely be more pertinent than those from completely different ones. In an extreme case, the task distribution might consist of samples that cannot even be directly labeled. This might apply in protein folding, where complex human proteins might make up the task distribution of interest, which would be too costly to crystallize, while the pool set might contain simpler proteins which could be more easily crystallized and their three-dimensional structure learned. Here, we will show that BALD can be actively counterproductive in cases like these, picking out the most obscure and least relevant inputs, potentially.

EPIG. To address BALD's shortcomings, we propose the expected predictive information gain (EPIG), an alternative acquisition function. We derive EPIG by returning to the foundational framework of Bayesian experimental design [Lindley, 1956], from which BALD itself is derived. Whereas BALD is the EIG in the model parameters, EPIG is the EIG in the model's predictions: it measures how much information the label of a candidate input is expected to provide about the label of a random target

input. While BALD favors global reductions in parameter uncertainty, EPIG favors only information that reduces downstream predictive uncertainty (Figure 7.1). Thus, EPIG allows us to directly seek improvements in predictive performance.

The randomness of the target input in EPIG is critical. We do not aim for predictive information gain on a particular input or set of inputs. Instead, the gain is in expectation with respect to a target input distribution. This can be chosen to be the same distribution that the pool of unlabeled inputs is drawn from, or it can be a distinct distribution that reflects a downstream task of interest.

We find that EPIG often produces notable gains in final predictive performance over BALD across a range of datasets and models. EPIG's gains are largest when the pool of unlabeled inputs contains a high proportion of irrelevant inputs with respect to the target input distribution. But its advantage still holds when the pool is directly drawn from this distribution. As such, it can provide a simple and effective drop-in replacement for BALD in many settings.

JEPIG. We also provide a preliminary examination of an alternative target input distribution acquisition function, which we name the joint expected predictive information gain (JEPIG). Like the EIG, JEPIG is still based around targeting information gain in the model parameters, but it discounts information that is not relevant to model's prediction for the target input distribution. Specifically, it discards from the EIG any information that will remain unknown after we would have acquired the labels for the target input distribution, on the basis that this information would not be helpful for prediction.

7.1 Shortfalls of BALD

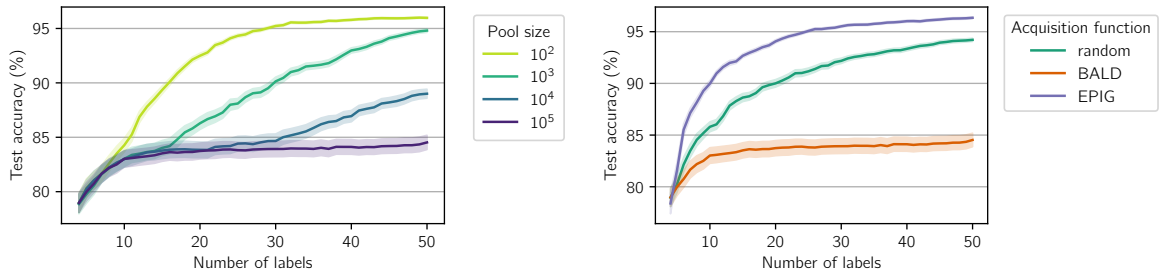
In this section, we highlight that BALD can be poorly suited to the prediction-oriented settings that constitute much of machine learning. We explain that this stems from the mismatch that can exist between parameter uncertainty and predictive uncertainty. Importantly, we find that BALD does not take the input distribution into account at all—it, thus, cannot target the predictive uncertainty of the inputs we actually want to make predictions on.

7.1.1 No Focus on Predictions

In statistics, it is common for the model parameters to be valued in their own right [Beck and Arnold, 1977; Blei et al., 2001; Fisher, 1925]. But in many machine-learning contexts, particularly the supervised settings where BALD is typically applied, the parameters are only valued insofar as they serve a prediction-oriented goal. We often, for example, seek the parameters that maximize the model's predictive performance on a test data distribution [Hastie et al., 2001]. This frequentist notion of success often remains our motivation even if we use a Bayesian approach to data acquisition and/or learning [Komaki, 1996; Snelson and Ghahramani, 2005].

7.1.2 No Notion of an Input Distribution

In order to reason about what information is relevant to prediction, we need some notion of the inputs on which we want to make predictions. Without this we have no mechanism to ensure the model we learn is well-suited to the task we care about. Our



(a) Pool Size Ablation **(b)** EPIG vs BALD vs Random

Figure 7.2: *Shortfalls of BALD, and how we fix them.* See Figure 7.1 for intuition and §7.3.4 for details. **(a)** BALD can fail catastrophically on big pools. A bigger pool typically contains more inputs with low density under the data-generating distribution. Often these inputs are of low relevance if the aim is to maximize expected predictive performance. BALD can nevertheless favor these inputs. **(b)** In contrast with BALD, EPIG deals effectively with a big pool (10^5 unlabeled inputs). BALD is overwhelming counterproductive even relative to random acquisition.

model could be highly effective on inputs from one region of input space but useless for typical samples from an input distribution of interest.

Appreciating the need to account for which inputs might arise at test time, it becomes clear why BALD can be problematic. BALD focuses on the model parameters in isolation, with no explicit connection to prediction. As such, it does not account for the distribution over inputs.

7.1.2.1 Failures under Real-World Data

BALD can be particularly problematic in the very settings that often motivate active learning: those where we have access to a large pool of unlabeled inputs whose relevance to some task of interest varies widely. In contrast with the carefully curated datasets often used in basic research, real-world data is often drawn from many sources of varying fidelity and relation to the task. Pools of web-scraped audio, images and text are canonical examples of this. Active learning ought to help deal with the mess by identifying only the most useful inputs to label. But BALD can in fact be worse than random acquisition in these settings, targeting obscure data that is not helpful for prediction.

The experiment presented in Figure 7.2(a) highlights this flaw. As we increase the size of the pool that BALD is maximized over, inputs of greater obscurity become more likely to be included in the pool, and BALD produces worse and worse predictive accuracy. This result is corroborated by the work of Karamcheti et al. [2021]. Focusing on visual-question-answering tasks, they found that BALD failed to outperform random acquisition when using uncurated pools, and that a substantial amount of curation was required before this shortfall could be overturned.

7.1.2.2 Failures *without* Distribution Shift

It might be tempting to just think of this problem with BALD as being analogous to the issues caused by train-test input-distribution shifts elsewhere in machine learning. But the problem is more deep-rooted than this: BALD has no notion of any input distribution in the first place. This is why increasing the size of the pool can induce

failures as in Figure 7.2(a), without any distribution shift or changes to the distribution that the pool inputs are drawn from. Distribution shift can cause additional problems for BALD, as the results in §7.3.3 show. But it is by no means a necessary condition for failure to occur.

7.1.3 Not All Information is Equal

In some models, such as linear models, parameters and predictions are tightly coupled, but more generally the coupling can be loose. This means that a reduction in parameter uncertainty typically might not yield a wholesale reduction in predictive uncertainty [Chaloner and Verdinelli, 1995a]. Deep neural networks, for instance, can have substantial redundancy in their parameters [Belkin et al., 2019], while Bayesian non-parametric models can be thought of as having an infinite number of parameters [Hjort et al., 2010]. When the coupling is loose, parameter uncertainty can be reduced without a corresponding reduction in predictive uncertainty on inputs of interest. In fact, it is possible to gain an infinite amount of parameter information while seeing an arbitrarily small reduction in predictive uncertainty.

Example 7.1 (Infinite Information Gain). Consider the supervised-learning problem depicted in Figure 7.1(a), where $x \in \mathbb{R}$ is an input, $y \in \mathbb{R}$ is a label, and we use a model consisting of a Gaussian likelihood function, $p(y|x, \mathbf{f}) = \mathcal{N}(\mathbf{f}(x), 1)$, and a zero-mean Gaussian-process prior, $\mathbf{f} \sim \mathrm{GP}(0, k)$, with covariance function $k(x, x') = \exp(-(x-x')^2/2l^2)$, where l is the length scale of the model [Williams and Rasmussen, 2006], e.g. $l = 2$. Suppose we are interested in making predictions $Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}$ in the region, e.g. $\boldsymbol{x}^{\text{eval}} \sim \mathcal{N}(2, (1/4)^2)$, and consider we are observing only odd integers x: $x_1, x_2, \ldots, x_M$ for some $M \in \mathbb{N}^+$ with predictions $Y_1, Y_2, \ldots, Y_M$. We only allow odd integers to avoid direct overlap with the region of interest as we decrease as the length scale (to make this example easier to reason about).

We will vary the length scale and the number of points M to show that BALD can become arbitrarily large while the predictive uncertainty of interest, $\mathrm{H}[Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}, \ldots]$, will change arbitrarily little.

BALD is the EIG between the predictions and GP functions $\mathbf{f} \sim GP(0, k)$ (as a non-parametric model):

$$\begin{aligned} \mathrm{I}[\mathbf{F}; Y_1, Y_2, \ldots, Y_M; \mid x_1, x_2, \ldots, x_M] &= \ldots \\ &= \mathrm{H}[Y_1, Y_2, \ldots, Y_M \mid x_1, x_2, \ldots, x_M] - \mathrm{H}[Y_1, Y_2, \ldots, Y_M \mid x_1, x_2, \ldots, x_M, \mathbf{F}]. \end{aligned} \tag{7.1}$$

The first term is the (joint) entropy of the predictions (incl. the Gaussian likelihood), and the second term is the (joint) entropy of the Gaussian likelihood itself. The entropy of the multivariate Gaussian distribution with covariance $\Sigma \in \mathbb{R}^{M \times M}$ is [Cover and Thomas, 2005]:

$$\mathrm{H}[\mathcal{N}(\mu, \Sigma)] = \frac{1}{2} \log \left((2\pi)^M \det \Sigma \right). \tag{7.2}$$

For the two terms, we have:

$$\mathrm{H}[Y_1, Y_2, \ldots, Y_M \mid x_1, x_2, \ldots, x_M] = \frac{1}{2} \log \left((2\pi)^M \det \left((k(x_i, x_j))_{ij} + \mathrm{Id}_M \right) \right), \tag{7.3}$$

$$\mathrm{H}[Y_1, Y_2, \ldots, Y_M \mid x_1, x_2, \ldots, x_M, \mathbf{F}] = \frac{1}{2} \log \left((2\pi)^M \det \left(\mathrm{Id}_M \right) \right). \tag{7.4}$$

We see that the $\log(2\pi)^M$ constants cancel out in both terms, and we end up with:

$$\begin{aligned}&\mathrm{I}[Y_1, Y_2, \ldots, Y_M; \mathbf{F} \mid x_1, x_2, \ldots, x_M] \qquad (7.5)\\ &\quad = \frac{1}{2}\log\det\left((k(x_i,x_j))_{ij} + \mathrm{Id}_M\right) - \frac{1}{2}\underbrace{\log\det \mathrm{Id}_M}_{=\log 1 = 0}\\ &\quad = \frac{1}{2}\log\det\left((k(x_i,x_j))_{ij} + \mathrm{Id}_M\right). \qquad (7.6)\end{aligned}$$

Now we can use the length scale: for $l \to 0$, we have $k(x_i, x_j) \to \mathbb{1}\{i = j\}$—the kernel matrix becomes an identity matrix—and thus, $\det\left((k(x_i,x_j))_{ij} + \mathrm{Id}_M\right) \to \det(2\mathrm{Id}_M) \to 2^M$. Similarly, we are interested in the reduction in the predictive uncertainty, $\mathrm{H}[Y^{\mathrm{eval}} \mid \boldsymbol{X}^{\mathrm{eval}}, \ldots]$:

$$\mathrm{H}[Y^{\mathrm{eval}} \mid \boldsymbol{X}^{\mathrm{eval}}] - \mathrm{H}[Y^{\mathrm{eval}} \mid \boldsymbol{X}^{\mathrm{eval}}, Y_1, x_1, Y_2, x_2, \ldots, Y_M, x_M]. \qquad (7.7)$$

With our eyes now well-trained for information quantities, we can see that this just the EIG of the predictive $Y^{\mathrm{eval}} \mid \boldsymbol{X}^{\mathrm{eval}}$ with respect to the samples $Y_1, Y_2, \ldots, Y_M \mid x_1, x_2, \ldots, x_M$:

$$\mathrm{I}[Y^{\mathrm{eval}}; Y_1, Y_2, \ldots, Y_M \mid \boldsymbol{X}^{\mathrm{eval}}, x_1, x_2, \ldots, x_M]. \qquad (7.8)$$

This is the expected predictive information gain, which we will introduce in §7.3. We can compute this quantity via $\mathrm{H}[A \mid B] = \mathrm{H}[A, B] - \mathrm{H}[B]$. Let us fix $\boldsymbol{x}^{\mathrm{eval}}$. The constant log terms cancel out again, and we have:

$$\begin{aligned}&\mathrm{I}[Y^{\mathrm{eval}}; Y_1, Y_2, \ldots, Y_M \mid \boldsymbol{x}^{\mathrm{eval}}, x_1, x_2, \ldots, x_M]\\ &\quad = \mathrm{H}[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}] + \mathrm{H}[Y_1, Y_2, \ldots, Y_M \mid x_1, x_2, \ldots, x_M] \qquad (7.9)\\ &\qquad - \mathrm{H}[Y^{\mathrm{eval}}, Y_1, Y_2, \ldots, Y_M \mid \boldsymbol{x}^{\mathrm{eval}}, x_1, x_2, \ldots, x_M]\\ &\quad = \frac{1}{2}\log\det\Big((k(\boldsymbol{x}^{\mathrm{eval}}, \boldsymbol{x}^{\mathrm{eval}})) + \mathrm{Id}_M\Big) + \frac{1}{2}\log\det\left((k(x_i,x_j))_{ij} + \mathrm{Id}_M\right) \qquad (7.10)\\ &\qquad - \frac{1}{2}\log\det\Big((k(x_i,x_j))_{i,j=\{*,1..M\}} + \mathrm{Id}_M\Big)\\ &\quad \overset{l\to 0}{\to} \frac{1}{2}\log 2 + \frac{1}{2}M\log 2 - \frac{1}{2}(M+1)\log 2 \qquad (7.11)\\ &\quad = 0. \qquad (7.12)\end{aligned}$$

Then, taking the expectation over $\boldsymbol{x}^{\mathrm{eval}}$, we have:

$$\begin{aligned}&\mathrm{I}[Y^{\mathrm{eval}}; Y_1, Y_2, \ldots, Y_M \mid \boldsymbol{X}^{\mathrm{eval}}, x_1, x_2, \ldots, x_M] \qquad (7.13)\\ &\quad = \mathbb{E}_{\mathrm{p}(\boldsymbol{x}^{\mathrm{eval}})}[\mathrm{I}[Y^{\mathrm{eval}}; Y_1, Y_2, \ldots, Y_M \mid \boldsymbol{x}^{\mathrm{eval}}, x_1, x_2, \ldots, x_M]]\\ &\quad \overset{l\to 0}{\to} \mathbb{E}_{\mathrm{p}(\boldsymbol{x}^{\mathrm{eval}})}[0] = 0. \qquad (7.14)\end{aligned}$$

Altogether, as $l \to 0$ and for $M \to \infty$, BALD diverges to infinity while the EIG in the prediction of interest converges to zero:

$$\mathrm{I}[\mathbf{F}; Y_1, Y_2, \ldots, Y_M \mid x_1, x_2, \ldots, x_M] \overset{l\to 0}{\to} \frac{M}{2}\log 2 \overset{M\to\infty}{\to} \infty, \qquad (7.15)$$

$$\mathrm{I}[Y^{\mathrm{eval}}; Y_1, Y_2, \ldots, Y_M \mid \boldsymbol{X}^{\mathrm{eval}}, x_1, x_2, \ldots, x_M] \overset{l\to 0}{\to} 0 \implies \mathrm{H}[Y^{\mathrm{eval}} \mid \boldsymbol{X}^{\mathrm{eval}}, \ldots] \approx \textit{const.} \qquad (7.16)$$

This example is a concrete demonstration that a high BALD score need not coincide with any reduction in the predictive uncertainty of interest. If the aim is to predict, then maximizing BALD is not guaranteed to help to any extent whatsoever.

7.2 Distribution-Aware Acquisition Functions

Motivated by BALD's weakness in prediction-oriented settings and the need for principled approaches, we return to the framework of Bayesian experimental design that underlies BALD, and derive two acquisition functions that we call the expected predictive information gain (EPIG) and the joint expected predictive information gain (JEPIG). Whereas BALD targets a reduction in parameter uncertainty, EPIG and JEPIG directly target a reduction in predictive uncertainty on inputs of interest.

7.2.1 Why not use Filtering Heuristics?

We might suppose we could just discard irrelevant data before deploying BALD. But this filtering process would require us to be able to determine each input's relevance at the outset of training, which is impractical in many cases. Even if we have access to a target input distribution, this on its own can be insufficient for judging relevance to a task of interest. A candidate input could have relatively low density under the target distribution but nevertheless share high-level features with a target input, such that the two inputs' labels are highly mutually informative. With high-dimensional inputs, it can also be surprisingly difficult to identify unrepresentative inputs purely through their density [Nalisnick et al., 2019]. Rather than trying to design an auxiliary process to mitigate BALD's problematic behavior, we seek an acquisition function that can automatically determine what is relevant.

7.2.2 Two Alternative Expected Information Gains

To be precise, when we talk about EIG and BALD, these are the *parameter EIG*, which does not focus on predictions and is not distribution-aware. But we can also look at different EIGs which focus on the predictions: target EIG or *predictive EIG*[1].

Evaluation Distribution. To reason about the predictions we are interested in, we need an explicit notion of the predictions we want to make with our model. We therefore introduce a random target input, $\boldsymbol{x}^{\text{eval}} \sim \text{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$, for unlabeled *evaluation samples* from the target input distribution and define our goal to be the confident prediction[2] of $y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}$.

Similar to other unlabeled samples in active learning, these samples could be either provided in a stream-based setting, where we can repeatedly draw new unlabeled samples i.i.d., or in a pool-based setting, where we only have a fixed reservoir of evaluation samples, which we call the *evaluation set* $\mathcal{D}^{\text{eval}}$. In the pool-based setting, we denote the *empirical* distribution of the evaluation set by $\text{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$, whereas in a stream-based setting, we could set $\text{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}}) \triangleq \text{p}_{\text{test}}(\boldsymbol{x}^{\text{eval}})$. This allows us to abstract away the exact setting in the following exposition.

Predictive EIG. Let us fix a single target input $\boldsymbol{x}^{\text{eval}} \sim \text{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$ with prediction $Y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}$, and consider the expected information gain given a single pool sample $\boldsymbol{x}^{\text{acq}} \in \mathcal{D}^{\text{pool}}$ with prediction $Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}$. The corresponding predictive EIG is, of course:

$$\text{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}]. \tag{7.17}$$

[1] Any naming scheme is going to be confusing and ambiguous given different prior art, so we might sadly add to this confusion.

[2] We still have to hope that by virtue of providing correct labels for training, confident predictions will also be likely correct predictions.

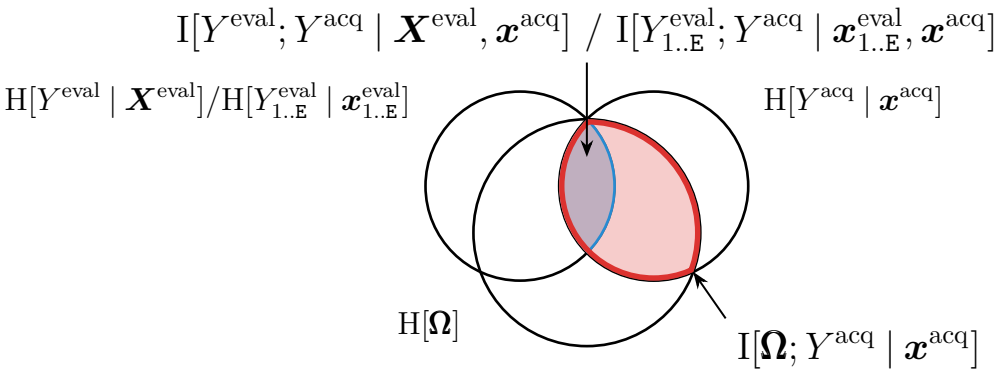


Figure 7.3: *Visualizing both EPIG* $\mathrm{I}[Y^{\mathrm{eval}}_{1..\mathtt{E}}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\mathrm{acq}}]$ *& JEPIG* $\mathrm{I}[Y^{\mathrm{eval}}; Y^{\mathrm{acq}} \mid \boldsymbol{X}^{\mathrm{eval}}, \boldsymbol{x}^{\mathrm{acq}}]$ *vs. EIG* $\mathrm{I}[\boldsymbol{\Omega}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}]$ *in the same I-Diagram.* MacKay's 'total information gain' for the EIG is a fitting term because we can immediately read off that it upper-bounds both EPIG and JEPIG.

The case when we want to take the whole distribution $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$ over $\boldsymbol{x}^{\mathrm{eval}}$ into account is slightly more complicated. There are two alternatives that we can examine:

- the *(mean) marginal predictive EIG*,

$$\mathrm{I}[Y^{\mathrm{eval}}; Y^{\mathrm{acq}} \mid \boldsymbol{X}^{\mathrm{eval}}, \boldsymbol{x}^{\mathrm{acq}}], \tag{7.18}$$

  where we take an expectation over $\boldsymbol{X}^{\mathrm{eval}}$, hence 'mean'; and
- the *joint predictive EIG*,

$$\mathrm{I}[Y^{\mathrm{eval}}_{1..\mathtt{E}}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\mathrm{acq}}], \tag{7.19}$$

  for an empirical evaluation set $\mathcal{D}^{\mathrm{eval}} = \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}$. (We will look at the general case below.)

We will see that the marginal predictive EIG is equivalent to the expected reduction in generalization loss if we were to acquire $\boldsymbol{x}^{\mathrm{acq}}$, while the joint predictive EIG is equivalent to performing Bayesian model selection on the pool set.

In the next sections, we will explore both using slightly different names, the joint predictive EIG as joint expected predictive information gain, and the (marginal) predictive EIG as expected predictive information gain. Sadly, there are several naming schemes. EPIG and JEPIG are mentioned in already published papers that spun out of this thesis, and while above names seem more fitting in retrospect, they also do not exactly match the ones from MacKay [1992b], who referred to them as 'mean marginal information gain' and 'joint information gain', respectively, and left out the rather important term 'predictive'. The parameter EIG was referred to as 'total information gain' in the same paper, which also makes sense: Figure 7.3 shows that the (parameter) EIG upper-bounds both EPIG and JEPIG. Hence, it is the total information gain, while EPIG and JEPIG only focus on the part of the information gain that is relevant for the predictions.

7.3 Expected Predictive Information Gain

To derive EPIG, we first consider the information gain (IG) in y^{eval} that results from conditioning on new data, $(\boldsymbol{x}, y)$ and fix $\boldsymbol{x}^{\mathrm{eval}} \sim \mathrm{p}(\boldsymbol{x}^{\mathrm{eval}})$:

$$\mathrm{I}[Y^{\mathrm{eval}}; y \mid \boldsymbol{x}^{\mathrm{eval}}, \boldsymbol{x}, \mathcal{D}^{\mathrm{train}}] = \mathrm{H}[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \mathcal{D}^{\mathrm{train}}] - \mathrm{H}[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, y, \boldsymbol{x}, \mathcal{D}^{\mathrm{train}}] \tag{7.24}$$

□ 7.1 – Expected Reduction in Generalization Loss

The same samples maximize EPIG and minimize the expected generalization loss [Roy and McCallum, 2001]. EPIG measures the expected reduction in the uncertainty in a model's predictions on target points $\boldsymbol{x}^{\text{eval}} \sim \mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$:

$$\begin{aligned}&\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}] = \\ &\quad \mathrm{H}[Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}, Y^{\text{acq}}_{1..\mathtt{K}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}]. \end{aligned} \tag{7.20}$$

Importantly, this objective is equivalent to minimizing the *expected generalization loss* under the model's predictions:

$$\begin{aligned}&\mathrm{H}[Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}, Y^{\text{acq}}_{1..\mathtt{K}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}] = \\ &\quad = \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})} \mathbb{E}_{\mathrm{p}(y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}})} \mathbb{E}_{\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, y^{\text{acq}}_{1..\mathtt{K}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}})} \mathcal{L}(\boldsymbol{x}^{\text{eval}}, y^{\text{eval}}), \end{aligned} \tag{7.21}$$

$$\mathcal{L}(\boldsymbol{x}^{\text{eval}}, y^{\text{eval}}) \triangleq -\log \mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, y^{\text{acq}}_{1..\mathtt{K}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}), \tag{7.22}$$

where we have marginalized over the model parameters

$$\boldsymbol{\omega} \sim \mathrm{p}(\boldsymbol{\omega} \mid y^{\text{acq}}_{1..\mathtt{K}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}). \tag{7.23}$$

$$= \mathrm{H}(\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, \mathcal{D}^{\text{train}})) - \mathrm{H}(\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, y, \boldsymbol{x}, \mathcal{D}^{\text{train}})) \tag{7.25}$$

where $\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, y, \boldsymbol{x}, \mathcal{D}^{\text{train}}) = \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega} \mid y, \boldsymbol{x}, \mathcal{D}^{\text{train}})}[\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, y, \boldsymbol{x}, \boldsymbol{\omega})]$. Note that this is a function of $\boldsymbol{x}^{\text{eval}}$ as well as $\boldsymbol{x}$ and y. Next we take an expectation over both the random target input, $\boldsymbol{x}^{\text{eval}}$, and the unknown label, y:

$$\mathrm{EPIG}(\boldsymbol{x}) = \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})\, \mathrm{p}(y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}})}[\mathrm{I}[Y^{\text{eval}}; y \mid \boldsymbol{x}^{\text{eval}}, \boldsymbol{x}, \mathcal{D}^{\text{train}}]] \tag{7.26}$$

$$= \mathrm{I}[Y^{\text{eval}}; Y \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}, \mathcal{D}^{\text{train}}], \tag{7.27}$$

where we were able to use our concise & unified notation for great benefit. Thus, we see that EPIG is the expected reduction in predictive uncertainty at a randomly sampled target input, $\boldsymbol{x}^{\text{eval}}$.

We can also take a frequentist perspective and relate it to expected reduction in prediction uncertainty, which is equivalent to the expected reduction in generalization loss (over $\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$) as we detail in **□ 7.1 Expected Reduction in Generalization Loss**.

There are other interpretations too. For example, we could write EPIG as an expected KL divergence. More interesting is that we can write EPIG as an expected information gain between $(\boldsymbol{X}^{\text{eval}}, Y^{\text{eval}})$ and $Y \mid \boldsymbol{x}$:

$$\mathrm{I}[(\boldsymbol{X}^{\text{eval}}, Y^{\text{eval}}); Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] = \underbrace{\mathrm{I}[\boldsymbol{X}^{\text{eval}}; Y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]}_{=0} + \mathrm{I}[Y^{\text{eval}}; Y \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}, \mathcal{D}^{\text{train}}]. \tag{7.28}$$

The first term on the right-hand side is 0 because $\boldsymbol{X}^{\text{eval}}$ and $Y \mid \boldsymbol{x}$ are independent. We can also write EPIG as difference between two EIGs:

$$\mathrm{I}[Y^{\text{eval}}; Y \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}, \mathcal{D}^{\text{train}}] \tag{7.29}$$

$$= \mathrm{I}[Y^{\mathrm{eval}}; Y \mid \boldsymbol{X}^{\mathrm{eval}}, \boldsymbol{x}, \mathcal{D}^{\mathrm{train}}] + \underbrace{\mathrm{I}[Y^{\mathrm{eval}}; Y \mid \boldsymbol{X}^{\mathrm{eval}}, \boldsymbol{x}, \boldsymbol{\Omega}, \mathcal{D}^{\mathrm{train}}]}_{=0} \tag{7.30}$$

$$= \mathrm{I}[Y^{\mathrm{eval}}; Y; \boldsymbol{\Omega} \mid \boldsymbol{X}^{\mathrm{eval}}, \boldsymbol{x}, \mathcal{D}^{\mathrm{train}}] \tag{7.31}$$

$$= \mathrm{I}[Y^{\mathrm{eval}}; \boldsymbol{\Omega} \mid \boldsymbol{X}^{\mathrm{eval}}, \mathcal{D}^{\mathrm{train}}] - \mathrm{I}[Y^{\mathrm{eval}}; \boldsymbol{\Omega} \mid \boldsymbol{X}^{\mathrm{eval}}, Y, \boldsymbol{x}, \mathcal{D}^{\mathrm{train}}]. \tag{7.32}$$

$\mathrm{I}[Y^{\mathrm{eval}}; Y \mid \boldsymbol{X}^{\mathrm{eval}}, \boldsymbol{x}, \boldsymbol{\Omega}, \mathcal{D}^{\mathrm{train}}]$ is zero because Y^{eval} and Y are independent given $\boldsymbol{\Omega}$. Thus, we can also view EPIG as the expected reduction in epistemic uncertainty for evaluation samples, Y^{eval}, given the model parameters, $\boldsymbol{\Omega}$.

7.3.1 Sampling Target Inputs

EPIG involves an expectation with respect to a target input distribution of evaluation samples, $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$. In practice, we estimate this expectation by Monte Carlo and so require a sampling mechanism.

In many active-learning settings an input distribution is implied by the existence of a pool of unlabeled inputs. There are cases where we know (or are happy to assume) the pool has been sampled from $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$. Alternatively we might be forced to assume this is the case: perhaps we know the pool is not sampled from $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$ but lack access to anything better. In these cases we can simply subsample from the pool to obtain samples of $\boldsymbol{x}^{\mathrm{eval}}$. Empirically we find that this can work well relative to acquisition with BALD (§7.3.3).

Another important case is where we have access to samples from $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$, but we cannot label them. Limits on the ability to acquire labels might arise due to privacy-related and other ethical concerns, geographical restrictions, the complexity of the labelling process for some inputs, or the presence of commercially sensitive information in some inputs. At the same time there might be a pool of inputs for which we have no labelling restrictions. In a case like this we can estimate EPIG using samples from $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$ while using only the pool as a source of candidates for labelling. Thus, we can target information gain in predictions on samples from $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$ without labelling those samples themselves.

A further scenario that we might encounter is a classification problem where the pool is representative of the target class-conditional input distribution but not the target marginal class distribution: that is, $\mathrm{p}_{\mathrm{pool}}(\boldsymbol{x}^{\mathrm{eval}} \mid y^{\mathrm{eval}}) = \mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}} \mid y^{\mathrm{eval}})$ but $\mathrm{p}_{\mathrm{pool}}(y^{\mathrm{eval}}) \neq \mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$. The pool might, for example, consist of uncurated web-scraped inputs from many more classes than those we care about. In this scenario it can often be the case that we know or can reasonably approximate the distribution over classes that we are targeting, $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$. With this we can approximately sample from $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$ using a combination of our model, $\mathrm{p}(y \mid \boldsymbol{x})$, and the pool. We first note that

$$\mathrm{p}_{\mathrm{pool}}(\boldsymbol{x}^{\mathrm{eval}} \mid y^{\mathrm{eval}}) = \frac{\mathrm{p}_{\mathrm{pool}}(\boldsymbol{x}^{\mathrm{eval}})\, \mathrm{p}_{\mathrm{pool}}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}})}{\int \mathrm{p}_{\mathrm{pool}}(\boldsymbol{x})\, \mathrm{p}_{\mathrm{pool}}(Y = y^{\mathrm{eval}} \mid x) dx}. \tag{7.33}$$

Then, using the fact that $\mathrm{p}_{\mathrm{pool}}(\boldsymbol{x}^{\mathrm{eval}} \mid y^{\mathrm{eval}}) = \mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}} \mid y^{\mathrm{eval}})$, we get

$$\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}}) = \sum_{y^{\mathrm{eval}}} \mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})\, \mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}} \mid y^{\mathrm{eval}}) \tag{7.34}$$

$$= \mathrm{p}_{\mathrm{pool}}(\boldsymbol{x}^{\mathrm{eval}}) \sum_{y^{\mathrm{eval}}} \frac{\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})\, \mathrm{p}_{\mathrm{pool}}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}})}{\int p_{\mathrm{pool}}(x) p_{\mathrm{pool}}(y = y^{\mathrm{eval}} | x) dx} \tag{7.35}$$

$$\approx \mathrm{p}_{\text{pool}}(\boldsymbol{x}^{\text{eval}}) \sum_{y^{\text{eval}}} \frac{\mathrm{p}_{\text{eval}}(y^{\text{eval}})\, \mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}})}{\frac{1}{N} \sum_{x \in \mathcal{D}_{\text{pool}}} \mathrm{p}(y = y^{\text{eval}} \mid x)} \tag{7.36}$$

$$= \mathrm{p}_{\text{pool}}(\boldsymbol{x}^{\text{eval}})\, w(\boldsymbol{x}^{\text{eval}}), \tag{7.37}$$

where we have approximated $\mathrm{p}_{\text{pool}}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}})$ with our model. Now we can approximately sample from $\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$ by subsampling inputs from the pool using a categorical distribution with probabilities $w(\boldsymbol{x}^{\text{eval}})/N$.

7.3.2 Estimation

The best way to estimate EPIG depends on the task and model of interest. In the empirical evaluations in this chapter we focus on classification problems and use models whose marginal and joint predictive distributions are not known in closed form. This leads us to use $\mathrm{EPIG}(\boldsymbol{x}) \approx$

$$\frac{1}{M} \sum_{j=1}^{M} \mathrm{D}_{\mathrm{KL}}(\hat{\mathrm{p}}(Y, Y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}_j^{\text{eval}}) \parallel \hat{\mathrm{p}}(Y \mid \boldsymbol{x})\, \hat{\mathrm{p}}(Y^{\text{eval}} \mid \boldsymbol{x}_j^{\text{eval}})), \tag{7.38}$$

where $\boldsymbol{x}_j^{\text{eval}} \sim \mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$, and $\hat{\mathrm{p}}$ denotes Monte Carlo approximations of the respective predictive distributions (see also Equations 1.24 and 1.25). Classification is an instance of where the required expectation over y and y^{eval} can be computed analytically, such that our only required estimation is from marginalizations over Θ.

If we cannot integrate over y and y^{eval} analytically, we can revert to nested Monte Carlo estimation [Rainforth et al., 2018]. For this we first note that, using Equation 1.25, we can sample $y, y^{\text{eval}} \sim \mathrm{p}(y, y^{\text{eval}} \mid x, \boldsymbol{x}^{\text{eval}})$ exactly by drawing a θ and then a y and y^{eval} conditioned on this θ. By also drawing samples for θ, we can then construct the estimator $\mathrm{EPIG}(\boldsymbol{x}) \approx$

$$\frac{1}{M} \sum_{j=1}^{M} \log \frac{K \sum_{i=1}^{K} \mathrm{p}(y_j \mid \boldsymbol{x}, \theta_i)\, \mathrm{p}(y_j^{\text{eval}} \mid \boldsymbol{x}_j^{\text{eval}}, \theta_i)}{\sum_{k=1}^{K} \mathrm{p}(y_j \mid \boldsymbol{x}, \theta_k) \sum_{k=1}^{K} \mathrm{p}(y_j^{\text{eval}} \mid \boldsymbol{x}_j^{\text{eval}}, \theta_k)}, \tag{7.39}$$

where $y_j, y_j^{\text{eval}} \sim \mathrm{p}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}_j^{\text{eval}})$, $\theta_i \sim \mathrm{p}(\theta)$, and $\boldsymbol{x}_j^{\text{eval}} \sim \mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$. Subject to some weak assumptions on p, this estimator converges as $K, M \to \infty$ [Rainforth et al., 2018].

The EPIG estimators in Equations 7.38 and 7.39 each have a total computational cost of $\mathcal{O}(MK)$. This is comparable to BALD estimation for regression problems. But it can be more expensive than BALD estimation for classification problems: BALD can be collapsed to a non-nested Monte Carlo estimation for an $\mathcal{O}(K)$ cost, but EPIG cannot.

Other possible estimation schemes include a variational approach inspired by Foster et al. [2019]. This is too expensive to be practically applicable in the settings we consider but could be useful elsewhere. See Appendix F.3 for details.

7.3.3 Empirical Validation

For consistency with existing work on active learning for prediction, our empirical evaluation of EPIG focuses on classification problems. Code for reproducing our results is available at `github.com/fbickfordsmith/epig`.

7.3.4 Synthetic Data (Figures 7.1, 7.2(a) and 7.2(b))

First we demonstrate the difference between BALD and EPIG in a setting that is easy to visualize and understand: binary classification with two-dimensional inputs.

Data. The first input distribution of interest, denoted $p_1(x)$ in Figure 7.1, is a bivariate Student's t distribution with $\nu = 5$ degrees of freedom, location $\mu = [0, 0]$ and scale matrix $\Sigma = 0.8I$. The second distribution, denoted $p_2(x)$ in Figure 7.1, is a scaled and shifted version of the first, with parameters $\nu = 5$, $\mu \approx [0.8, 0.9]$ and $\Sigma = 0.4I$. This serves to illustrate in Figure 7.1 how EPIG's value changes with the target input distribution; it is not used elsewhere. The conditional label distribution is defined as $p(y = 1|x) = \Phi(20(\tanh(2x_{[1]}) - x_{[2]}))$, where $x_{[i]}$ denotes the component of input x in dimension i, and Φ is the cumulative distribution function of the standard normal distribution. For the training data in Figure 7.1, we sample ten input-label pairs, $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{10}$, where $x_i, y_i \sim p(y|x)p_1(x)$. Likewise, we sample $\mathcal{D}_{\text{test}} = \{(x_i, y_i)\}_{i=1}^{10,000}$ for evaluating the model's performance in active learning.

Model and Training. We use a model with a probit likelihood function, $\mathrm{p}(y = 1 \mid x, \theta) = \Phi(\theta(x))$, where Φ is defined as above, and a Gaussian-process prior, $\theta \sim \mathrm{GP}(0, k)$, where $k(x, x') = 10 \cdot \exp\left(-\|x - x'\|^2/2\right)$. The posterior over latent-function values cannot be computed exactly, so we optimize an approximation to it using variational inference [Hensman et al., 2015]. To do this we run 10,000 steps of full-batch gradient descent using a learning rate of 0.005 and a momentum factor of 0.95.

Active Learning. We initialize the training dataset, $\mathcal{D}_{\text{train}}$, with two randomly sampled inputs from each class. Thereafter, we run the active-learning loop described in §1.2.4 until a budget of 50 labels is used up. We acquire data using three acquisition functions: random, BALD and EPIG. Random acquisition involves sampling uniformly from the pool without replacement. We estimate BALD using Equation F.6 and EPIG using Equation 7.38, in both cases drawing 5,000 samples from the model's approximate posterior. For EPIG we sample $\boldsymbol{x}^{\text{eval}} \sim p_1(\boldsymbol{x}^{\text{eval}})$. After each time the model is trained, we evaluate its predictive accuracy on $\mathcal{D}_{\text{test}}$ as defined above. Using a different random-number-generator seed each time, we run active learning with each acquisition function 100 times. We report the test accuracy (mean ± standard error) as a function of the size of $\mathcal{D}_{\text{train}}$.

Discussion Figure 7.2(b) shows a striking gap between BALD and EPIG in active learning. Figures 7.1 and 7.2(a) provide some intuition about the underlying cause of this disparity: BALD has a tendency to acquire labels at the extrema of the input space, regardless of their relevance to the predictive task of interest.

7.3.5 UCI Data (Figure 7.4)

Next we compare BALD and EPIG in a broader range of settings. We use problems drawn from a repository maintained at UC Irvine (UCI; Dua and Graff, 2017), which has been widely used as a data source in past work on Bayesian methods [Gal and Ghahramani, 2016a; Lakshminarayanan et al., 2017; Sun et al., 2018; Zhang et al., 2018]. The problems we use vary in terms of the number of classes, the input dimension and any divergence between the pool and target data distributions. We assume knowledge of $\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$ when estimating EPIG but note that this assumption has little significance if $\mathrm{p}_{\text{pool}}(\boldsymbol{x})$ and $\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$ match, which is true for two out of the three problems.

Data. We use three classification datasets from the UCI repository, each with a different number of classes, C, and input dimension, D: Magic ($C = 2$, $D = 11$), Satellite ($C = 6$, $D = 36$) and Vowels ($C = 11$, $D = 10$). The inputs are telescope readings in Magic, satellite images in Satellite and speech recordings in Vowels. Magic

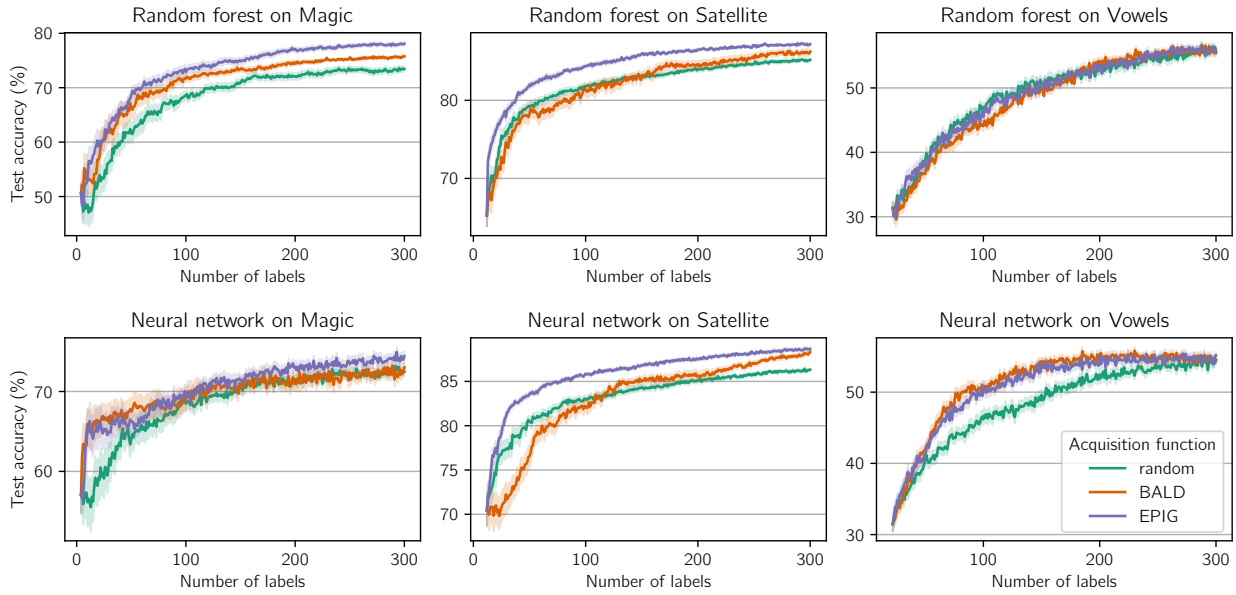


Figure 7.4: EPIG outperforms or matches BALD across three standard classification problems from the UCI machine-learning repository (Magic, Satellite and Vowels) and two models (random forest and neural network). See §7.3.5 for details.

is interesting because it serves as a natural instance of a mismatch between pool and target distributions (see Appendix F.4.1).

Models and Training. We use two different models. The first is a random forest [Breiman, 2001]. To emphasize that EPIG works with an off-the-shelf setup, we use the Scikit-learn [Pedregosa et al., 2011b] implementation with its default parameters. The second model is a dropout-enabled fully connected neural network with three hidden layers and a softmax output layer. A dropout rate of 0.1 is used during both training and testing. Training the neural network consists of running up to 50,000 steps of full-batch gradient descent using a learning rate of 0.1. We use a loss function consisting of the negative log likelihood (NLL) of the training data combined with an l_2 regularizer (with coefficient 0.0001) on the model parameters. To mitigate overfitting we use early stopping: we track the model's NLL on a small validation set (approximately 20% of the size of the training-label budget) and stop training if this NLL does not decrease for more than 10,000 consecutive steps. We then restore the model parameters to the configuration that achieved the lowest validation-set NLL.

Active Learning. We use largely the same setup as described in §7.3.4. Here the label budget is 300, and we run active learning 20 times with different seeds. We use the same BALD and EPIG estimators as before, treating each tree in the random forest as a different θ value, and treating each stochastic forward pass through the neural network (we compute 100 of them) as corresponding to a different θ value. To estimate EPIG we sample $\boldsymbol{x}^{\text{eval}}$ from a set of inputs designed to be representative of $\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$.

Discussion. Figure 7.4 shows EPIG performing convincingly better than BALD in some cases while matching it in others. These results provide broader validation of EPIG, complementing the results in Figure 7.2(b).

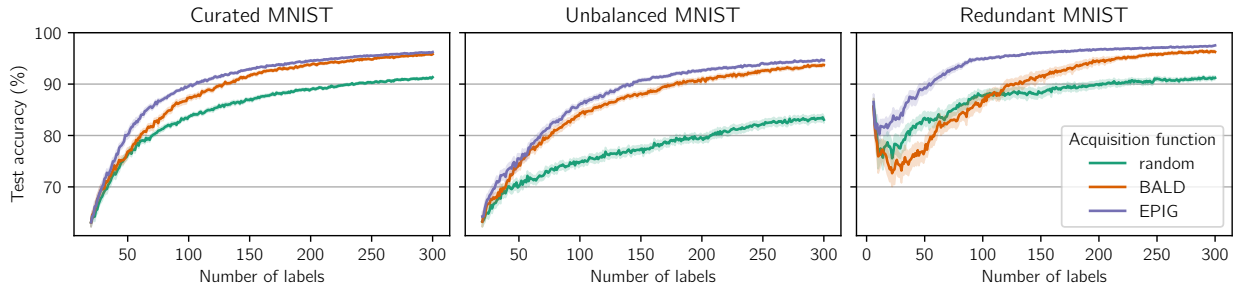


Figure 7.5: EPIG outperforms BALD across three image-classification settings. Curated MNIST reflects the data often used in academic research. The pool and target input distributions, $\mathrm{p}_{\mathrm{pool}}(\boldsymbol{x})$ and $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$ match; the marginal class distributions, $\mathrm{p}_{\mathrm{pool}}(y)$ and $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$, are uniform. Unbalanced MNIST is a step closer to real-world data. While $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$ remains uniform, $\mathrm{p}_{\mathrm{pool}}(y)$ is non-uniform: the pool contains more inputs from some classes than others. Redundant MNIST simulates a separate practical problem. Whereas $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$ only has nonzero mass on two classes of interest, $\mathrm{p}_{\mathrm{pool}}(y)$ has substantial mass across all classes. See §7.3.6 for details.

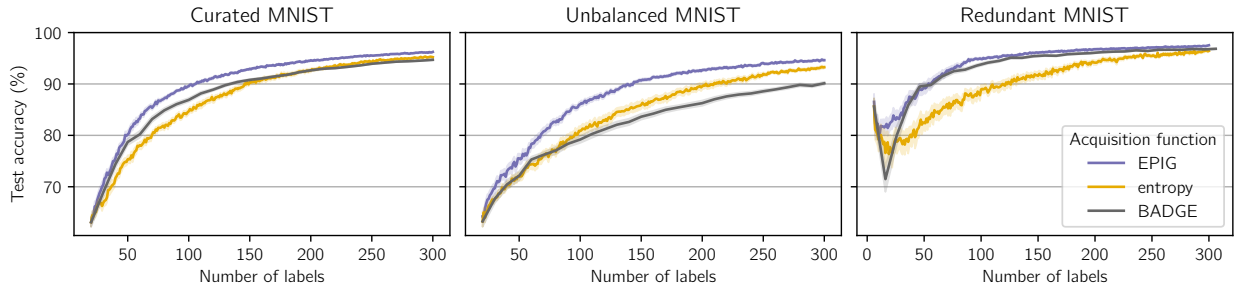


Figure 7.6: EPIG outperforms two acquisition functions popularly used as baselines in the active-learning literature. The first is the model's predictive entropy, $\mathrm{H}(\mathrm{p}(y \mid y))$ [Settles and Craven, 2008]. The second is BADGE [Ash et al., 2020]. Calculating BADGE involves computing a gradient-based embedding for each candidate input in the pool and then applying k-means++ initialization [Arthur and Vassilvitskii, 2007] in embedding space to select a diverse batch of inputs for labelling. We acquire 10 labels at a time with BADGE.

7.3.6 MNIST Data (Figures 7.5 to 7.7)

Finally, we evaluate BALD and EPIG in settings intended to capture challenges that occur when applying deep neural networks to high-dimensional inputs. Our starting point is the MNIST dataset [LeCun et al., 1998], in which each input is an image of a handwritten number between 0 and 9. This dataset has been widely used in related work on Bayesian active learning with deep neural networks [Beluch et al., 2018; Gal et al., 2017; Jeon, 2020; Lee and Kim, 2019; Tran et al., 2019]. We construct three settings based on this dataset, each corresponding to a different practical scenario: Curated MNIST, Unbalanced MNIST and Redundant MNIST.

As well as investigating how BALD and EPIG perform across these settings, we seek to understand the effect on EPIG of varying the amount of knowledge we have of the target data distribution, $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$. To this end we assume we know this for one set of runs (Figure 7.5) and then relax this assumption for another set (Figure 7.7).

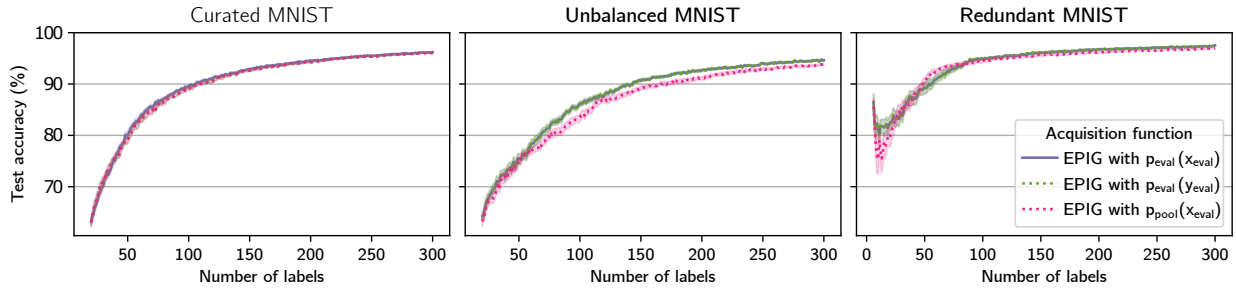


Figure 7.7: Even without knowledge of the target input distribution, $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$, EPIG retains its strong performance on Curated MNIST, Unbalanced MNIST and Redundant MNIST. "EPIG with $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$" assumes exact samples from $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$, as in Figure 7.5. "EPIG with $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$" corresponds to using the approximate-sampling scheme outlined in §7.3.1, using knowledge of $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$. "EPIG with $\mathrm{p}_{\mathrm{pool}}(\boldsymbol{x}^{\mathrm{eval}})$" corresponds to using samples from the pool as a proxy for $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$. See §7.3.6 for details.

Data. Curated MNIST is intended to reflect the data often used in academic machine-learning research. The pool and target class distributions, $\mathrm{p}_{\mathrm{pool}}(y)$ and $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$, are both uniform over all 10 classes. In terms of class distributions, this effectively represents a worst-case scenario for active learning relative to random acquisition. Given matching class-conditional input distributions, namely $\mathrm{p}_{\mathrm{pool}}(\boldsymbol{x}^{\mathrm{eval}} \mid y^{\mathrm{eval}}) = \mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}} \mid y^{\mathrm{eval}})$, uniformly sampling from the pool input distribution, $\mathrm{p}_{\mathrm{pool}}(\boldsymbol{x})$, is equivalent to uniformly sampling from the target input distribution, $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$. Thus, random acquisition is a strong baseline in this setting.

Unbalanced MNIST is a step closer to real-world data. We might expect $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$ to be uniform—that is, the task of interest might involve classifying examples in equal proportion from each class—but it could be difficult to curate a pool that is similarly uniform in its class distribution. To reflect this we consider a case with a non-uniform $\mathrm{p}_{\mathrm{pool}}(()y)$: classes 0 to 4 each have probability $1/55$ and classes 5 to 9 each have probability $10/55$.

Redundant MNIST captures a separate practical problem that occurs, for instance, when using web-scraped data. The pool might contain inputs from many more classes than we want to focus on in the predictive task of interest. To simulate this we suppose that the task involves classifying just images of 1s and 7s, occurring in equal proportion—that is, $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$ places probability mass of $1/2$ on class 1, $1/2$ on class 7, and 0 on all other classes—while $\mathrm{p}_{\mathrm{pool}}(y)$ is uniform over all 10 classes. If the acquisition function selects an input from a class other than 1 and 7, the labelling function produces a "neither" label. Thus, we have three-way classification during training: 1 vs 7 vs neither.

Model and Training. For both runs we use the same dropout-enabled convolutional neural network we already used in §4. The dropout rate here is 0.5. Training is similar to as described in §7.3.5, except that the learning rate is 0.01 and early stopping triggers after 5,000 consecutive steps of non-decreasing validation-set NLL.

Active Learning. Initially we retain the setup described in §7.3.5, with $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$ known (Figure 7.5). Then we investigate the sensitivity of EPIG to removing full access to $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$, focusing on two different settings (Figure 7.7). In one setting, we assume knowledge of $\mathrm{p}_{\mathrm{eval}}(y^{\mathrm{eval}})$ and use the resampling technique discussed in §7.3.1. In the other, we simply sample target inputs from the pool: $\boldsymbol{x}^{\mathrm{eval}} \sim \mathrm{p}_{\mathrm{pool}}(()\boldsymbol{x}^{\mathrm{eval}})$.

□ 7.2 – Expected Reduction of Epistemic Uncertainty

By rewriting the joint expected predictive information gain (JEPIG) as a triple mutual information which takes into account the model parameters we can rephrase it as a difference of two (parameter) EIG terms (dropping $\mathcal{D}^{\text{train}}$ for clarity):

$$\begin{aligned}
&\mathrm{I}[Y_{1..\mathtt{E}}^{\text{eval}}; Y_{1..\mathtt{K}}^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}] \\
&\quad = \mathrm{I}[Y_{1..\mathtt{E}}^{\text{eval}}; Y_{1..\mathtt{K}}^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}] - \underbrace{\mathrm{I}[Y_{1..\mathtt{E}}^{\text{eval}}; Y_{1..\mathtt{K}}^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{\Omega}]}_{=0} && (7.40) \\
&\quad = \mathrm{I}[Y_{1..\mathtt{E}}^{\text{eval}}; Y_{1..\mathtt{K}}^{\text{acq}}; \boldsymbol{\Omega} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}] && (7.41) \\
&\quad = \underbrace{\mathrm{I}[Y_{1..\mathtt{K}}^{\text{acq}}; \boldsymbol{\Omega} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}]}_{(5)} - \underbrace{\mathrm{I}[Y_{1..\mathtt{K}}^{\text{acq}}; \boldsymbol{\Omega} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, Y_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}]}_{(6)}. && (7.42)
\end{aligned}$$

Since BALD is known to measure epistemic uncertainty [Smith and Gal, 2018], we can take another look at these two EIGs: Intuitively, the first EIG (5) is large when the model has high epistemic uncertainty about its prediction at $\boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}$, and learning the true label would thus be informative for the model parameters, while the second EIG (6) captures the epistemic uncertainty about the model's prediction at $\boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}$ assuming we had obtained labels for evaluation samples. This second term is small when $\boldsymbol{x}$ is similar to drawn evaluation samples and the model can explain it well given the pseudo-labels. Thus, JEPIG is large when the first term is large and the second term is small, so learning $\boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}$ is informative for the model, and $\boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}$ is similar to evaluation samples. In reverse, JEPIG is small, when knowing about evaluation samples makes no difference for the epistemic uncertainty of the acquisition candidates, which means that they are unrelated.

Discussion. Figure 7.5 shows EPIG again outperforming BALD and random across all three dataset variants when given access to $\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$. (EPIG additionally beats predictive entropy [Settles and Craven, 2008] and BADGE [Ash et al., 2020], acquisition functions commonly studied in the active-learning literature, as shown in **??**.) EPIG's advantage over BALD is appreciable on Curated MNIST and Unbalanced MNIST. But it is emphatic on Redundant MNIST. This suggests EPIG is particularly useful when working with highly diverse pools.

Figure 7.7 shows the even more impressive result that EPIG retains its strong performance even when no access to $\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$ is assumed. We thus see that EPIG provides a good degree of robustness in its performance to the level of knowledge about the target data distribution.

7.4 Joint Expected Predictive Information Gain

In this section, we examine *joint expected predictive information gain (JEPIG)*,

$$\mathrm{I}[Y_{1..\mathtt{E}}^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}], \tag{7.43}$$

in more detail. As explained in **□ 7.2 Expected Reduction of Epistemic Uncertainty**, JEPIG also has an intuitive explanation as expected reduction in epistemic uncertainty

when taking into account evaluation samples.

First, we will show that we can take the limit of the number of evaluation set samples to model $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$. Then, we connect the criterion to Bayesian model selection and the marginal log likelihood. Finally, we contrast JEPIG with BALD and EPIG and present a practical reason why JEPIG might be easier to compute in the batch acquisition setting. We report some early results on the performance of JEPIG in §7.4.4.

7.4.1 JEPIG for $\mathtt{E} \to \infty$

If we do not have a finite number of evaluation samples but a distribution, we might want to take the limit to take into account the full distribution. For countable many $\boldsymbol{x}^{\mathrm{eval}}$, we can easily show that the limit exist.

Proposition 7.1. *The following limit exists for countable* $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$*:*

$$\lim_{\mathtt{E}\to\infty} \mathrm{I}[Y^{\mathrm{eval}}_{1..\mathtt{E}}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\mathrm{acq}}]. \tag{7.44}$$

Proof. Fix $\mathtt{E} > 1$, and fix the order of $\boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}$. JEPIG for fixed $\mathtt{E}$ is upper-bounded by the predictive entropy of the acquisition batch:

$$\mathrm{I}[Y^{\mathrm{eval}}_{1..\mathtt{E}}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\mathrm{acq}}] \leq \mathrm{H}[Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}], \tag{7.45}$$

and it is non-decreasing:

$$\begin{aligned}&\mathrm{I}[Y^{\mathrm{eval}}_{1..\mathtt{E}}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\mathrm{acq}}] \\ &\quad = \mathrm{I}[Y^{\mathrm{eval}}_{1..\mathtt{E}-1}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}-1}, \boldsymbol{x}^{\mathrm{acq}}] + \underbrace{\mathrm{I}[Y^{\mathrm{eval}}_{\mathtt{E}}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{\mathtt{E}}, Y^{\mathrm{eval}}_{1..\mathtt{E}-1}, \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}-1}, \boldsymbol{x}^{\mathrm{acq}}]}_{\geq 0}.\end{aligned} \tag{7.46}$$

Hence, the limit exists. However, it is not clear if the order of the evaluation samples matters.

To show that the limits are equal for arbitrary orderings of the evaluation, let's first fix a natural ordering $(1, 2, \ldots)$ and define:

$$I(n) \triangleq \mathrm{I}[Y^{\mathrm{eval}}_{1..n}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..n}, \boldsymbol{x}^{\mathrm{acq}}]. \tag{7.47}$$

Let $(a(1), a(2), \ldots)$ denote another arbitrary ordering, and let $I^a(n)$ denote the respective truncated JEPIG term:

$$I^a(n) \triangleq \mathrm{I}[Y^{\mathrm{eval}}_{a(1)..a(n)}; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{eval}}_{a(1)..a(n)}, \boldsymbol{x}^{\mathrm{acq}}]. \tag{7.48}$$

Finally, let's denote the limits for $n \to \infty$ as I and I^a, respectively, which exist according to the argument above. We can now show that $I = I^a$ for all orderings a.

Fix $\epsilon \geq 0$. Then, $\exists n \forall k \geq n : |I^a(k) - I^a(k)| \leq \epsilon$ because of the limit. For any n, we choose an M such that $\{1, \ldots, M\} \supseteq \{a(1), \ldots, a(n)\}$—we can simply take $M \triangleq \max(a(1), \ldots, a(n))$. Then, following the same argument as in Equation 7.46:

$$I(K) \geq I^a(n) \quad \forall K \geq M. \tag{7.49}$$

Further, $\forall K \geq M$, we can choose an $N(K)$, such that $\{a(1), \ldots, a(N)\} \supseteq \{1, \ldots, K\}$—as a is an ordering, it is bijective, and we can set $N(K) \triangleq \max(a^{-1}(1), \ldots, a^{-1}(K))$—and we obtain:

$$I^a \geq I^a(N(K)) \geq I(K). \tag{7.50}$$

Hence, we can sandwich $I(K)$ between $I^a(n)$ and I^a:

$$I^a \geq I(K) \geq I^a(n). \tag{7.51}$$

This concludes the proof because for all $K \geq M$:

$$|I^a - I(K)| \leq |I^a - I^a(n)| \leq \epsilon, \tag{7.52}$$

and we obtain the limit $I(K) \to I^a$. As already $I(K) \to I$, we have $I^a = I$. □

We conjecture that the limit also exists for continuous $\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$.

7.4.2 JEPIG as Bayesian Model Selection

JEPIG can also be viewed as a reduction in expected marginal log likelihood for the evaluation set: how much does acquiring a label for an acquisition sample improve the model's ability to adapt to the evaluation set? We can view this as Bayesian model selection:

In *Bayesian model selection*, for a set of models M_i and a random variable $\mathcal{M}$ that represents the chosen model, we are interested in inferring $\mathrm{p}(\mathcal{M} \mid \mathcal{D}^{\mathrm{train}})$. This quantifies the model that is best for our purposes. Conventionally, the marginal likelihood $\mathrm{p}(\mathcal{D}^{\mathrm{train}} \mid \mathcal{M})$ is maximized, which assumes a uniform prior distribution over $\mathcal{M}$ [MacKay, 2003]. The marginal likelihood does not actually measure the performance of the fully trained model but its generalization 'speed' [Lyle et al., 2020]. To measure the performance of a model, we can use cross-validation, which can be cast into a Bayesian setting [Fong and Holmes, 2020]. This was also explored in [Lotfi et al., 2022] (but see also Kirsch [2022] for a critical review). We explore aspects of this further in §6.

When $\mathcal{D}^{\mathrm{eval}} \subseteq \mathcal{D}^{\mathrm{pool}}$, maximizing JEPIG performs Bayesian model selection towards selecting the best acquisition batch in expectation: we precisely perform Bayesian model selection in the above sense by trying to find the acquisition batch that will help the model best adapt to future data points (assuming we aim to acquire all pool points eventually.) Obviously, this is not true when the evaluation set is disjoint from the pool.

7.4.3 JEPIG & EPIG vs. BALD

From Figure 7.3, we already know that BALD upper-bounds both JEPIG and EPIG. Let us examine this in more detail. From Equation 7.42 from the intuition box for JEPIG, we also see that we have

$$\begin{aligned} &\mathrm{I}[Y^{\mathrm{eval}}_{1..\mathtt{E}}; Y^{\mathrm{acq}}_{1..\mathtt{K}}; \boldsymbol{\Omega} \mid \boldsymbol{x}^{\mathrm{acq}}_{1..\mathtt{K}}, \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\mathrm{train}}] \\ &\quad \leq \mathrm{I}[Y^{\mathrm{acq}}_{1..\mathtt{K}}; \boldsymbol{\Omega} \mid \boldsymbol{x}^{\mathrm{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\mathrm{train}}], \end{aligned} \tag{7.53}$$

that is, JEPIG is upper-bounded by BALD—a two-term mutual information is always non-negative—and JEPIG is equivalent to BALD exactly when $\mathrm{I}[Y^{\mathrm{acq}}_{1..\mathtt{K}}; \boldsymbol{\Omega} \mid \boldsymbol{x}^{\mathrm{acq}}_{1..\mathtt{K}}, Y^{\mathrm{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\mathrm{train}}]$ is zero. This is the case when either there is no uncertainty about the parameters left, or when the distribution of $Y^{\mathrm{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\mathrm{acq}}_{1..\mathtt{K}}$ is fully determined by conditioning the posterior on $y^{\mathrm{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}} \sim \mathrm{p}(y^{\mathrm{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\mathrm{train}})$. When MacKay [1992b] analyzed JEPIG, they stated that it was equivalent to BALD because in the specific context of that work, using Bayesian linear regression with simple homoscedastic noise, it was.

BALD and JEPIG will trivially be equal when $\boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}} \subseteq \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}$. In reverse, for non-parametric models, this will trivially not be the case when $\boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}$ is not 'near' the evaluation set. Intuitively, JEPIG is different from BALD exactly when BALD fails: for outlier pool samples which are not similar to the test–time distribution, in which case JEPIG will tend towards 0 as the two terms cancel out.

Finally, a qualitative result:

Proposition 7.2. *EPIG lower-bounds 'averaged' JEPIG:*

$$\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] \leq {}^{1}\!/_{\mathtt{E}}\, \mathrm{I}[Y^{\text{eval}}_{1..\mathtt{E}}; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\text{acq}}] + c_{\text{eval}}, \tag{7.54}$$

up to an additive constant (c_{eval}) that only depends on the evaluation samples and is independent of $\boldsymbol{x}^{\text{acq}}$. The inequality gap is the total correlation:

$${}^{1}\!/_{\mathtt{E}}\, \mathrm{TC}[Y^{\text{eval}}_1; \ldots; Y^{\text{eval}}_{\mathtt{E}} \mid \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}] \tag{7.55}$$

We have equality when it is zero, that is when the predictions on the evaluation set are independent (given the acquisition samples).

See Appendix F.5 for the formal proof. This result might seem unintuitive given that we have the opposite for BatchBALD in §4. However, it is important to notice that the redundancy that JEPIG removes is in the evaluation set. In the case of BatchBALD, the redundancy is in the acquisition set. Hence, the different direction of the inequality sign.

Batch Acquisition using EPIG. A practical reason for examining JEPIG is batch active learning with EPIG. While the expectation in EPIG can be evaluated for individual acquisition, the batch setting is computationally more complex. Following §4, for batch acquisition, we need to maximize $\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}]$. Unlike the expected information gain, this term is not submodular. However, as the global subset problem is not feasible, we will examine the case of using greedy selection nonetheless. Computing $\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}]$ requires estimating a joint density in $i{+}1$ many variables for each $\boldsymbol{x}^{\text{eval}}$ sample for the i-th element in the acquisition batch: $\mathrm{p}(y^{\text{eval}}, y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}})$. Overall, for an acquisition batch of size B, this requires $O(|\mathcal{D}^{\text{eval}}|\,|\mathcal{D}^{\text{pool}}|^B)$ many joint densities with 2 to $B+1$ many variables compared to $O(|\mathcal{D}^{\text{pool}}|^B)$ for BatchBALD with 1 to B many variables. Unfortunately, BatchBALD has already been found to be computationally intractable for larger acquisition batch sizes in practice as noted in §4 and 5, and here we consider an additional variable by default, while we also have to evaluate this term for many $\boldsymbol{x}^{\text{eval}}$. This does not spark joy.

Evaluation of JEPIG. We could maximize $\mathrm{I}[Y^{\text{eval}}_{1..\mathtt{E}}; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}]$ via

$$\begin{aligned}
&\mathrm{I}[Y^{\text{eval}}_{1..\mathtt{E}}; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] = \\
&\quad = \mathrm{H}[Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] \\
&\qquad - \mathrm{H}[Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, Y^{\text{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\text{train}}].
\end{aligned} \tag{7.56}$$

Note that the second term is expensive to evaluate:

$$\begin{aligned}
&\mathrm{H}[Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, Y^{\text{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\text{train}}] = \\
&\quad = \mathbb{E}_{\mathrm{p}(y^{\text{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\text{train}})}\, \mathrm{H}[Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, y^{\text{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\text{train}}].
\end{aligned} \tag{7.57}$$

That is, we need to train a model on $\mathcal{D}^{\text{train}}$ then sample joint label predictions $y^{\text{eval}}_{1..\mathtt{E}}$ for the evaluation set $\boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}$ using the model, and then, for each such sampled joint

prediction, we evaluate the conditional entropy $\mathrm{H}[Y^{\text{acq}}_{1..K} \mid \boldsymbol{x}^{\text{acq}}_{1..K}, \boldsymbol{\Omega}']$ using new $\boldsymbol{\omega}' \sim \mathrm{p}(\boldsymbol{\omega} \mid y^{\text{eval}}_{1..E}, \boldsymbol{x}^{\text{eval}}_{1..E}, \mathcal{D}^{\text{train}})$, which requires additional training using the labeled training set augmented with the sampled labels $\mathcal{D}^{\text{train}} \cup \{y^{\text{eval}}_i, \boldsymbol{x}^{\text{eval}}_i\}_i$.

In appendix §F.6, we present a computationally tractable approximation for this term using '*self-distillation*', where we train a new model on predictions for evaluation samples from the current model trained on $\mathcal{D}^{\text{train}}$. On a high-level, this yields a model that fulfills the intuitions presented in □ **7.2 Expected Reduction of Epistemic Uncertainty**. We refer to the appendix for more details.

7.4.4 Empirical Validation

We evaluate the performance of JEPIG using a form of self-distillation described in appendix §F.6 in regular active learning and under distribution shift. Moreover, we provide an ablation with different evaluation set sizes. We use approximate BNNs based on MC dropout, see §1.2.2.

Setup. We compare EPIG and JEPIG with different acquisition sizes to BALD, either using the top-k of individual scores [Gal and Ghahramani, 2016a], the batch version BatchBALD (§4)—which is equivalent to the previous for individual acquisition—or SoftmaxBALD (§5) for larger acquisition batch sizes. SoftmaxBALD samples without replacement from the pool set using the Softmax of the acquisition scores with temperature 8.

On MNIST and MNISTx2 (Repeated-MNIST), we use a LeNet-5 model [LeCun et al., 1998], which we train as described in §4.

For CIFAR-10 [Krizhevsky, 2009], we use a ResNet18 model [He et al., 2016] which was modified as described in §4 to add MC dropout to the classifier head and also follows the described training regime. We train with an acquisition batch size of 250 and an initial training set size of 1000. We use MC dropout models with 100 dropout samples when computing the acquisition scores.

Performance on Regular Active Learning. We evaluate whether ignoring the test–time input distribution has a detrimental effect on BALD even when the pool set distribution matches the test distribution.

For this, we compare BALD and JEPIG in Figure 7.10 on MNISTx2 and EPIG and JEPIG in Figure 7.8 on MNIST. In the regular active learning setup, there is no distribution shift between the pool set and test set, so we use the whole unlabeled pool set as evaluation set.

Both in the top-k and the batch variant, EPIG and JEPIG outperform BALD on MNISTx2 (and also MNIST, not shown). On CIFAR-10, JEPIG also outperforms SoftmaxBALD, as depicted in Figure 7.9.

However, why does JEPIG outperform EPIG? We hypothesize that this is because EPIG takes an expectation over the evaluation set using individual points which by itself might be myopic. It might work well for simple models, but taking the whole evaluation set into account and retraining the model might allow for learning better abstraction in deep models, which might not be the case for EPIG.

Performance on Active Learning under Distribution Shift with MNIST and FashionMNIST. We want to evaluate how BALD and JEPIG behave under distribution shift, that is when pool set and test distribution do not match. For this, we add junk out-of-distribution data to the pool set. In this experiment, the pool set contains MNIST and FashionMNIST [Xiao et al., 2017] while the test set contains

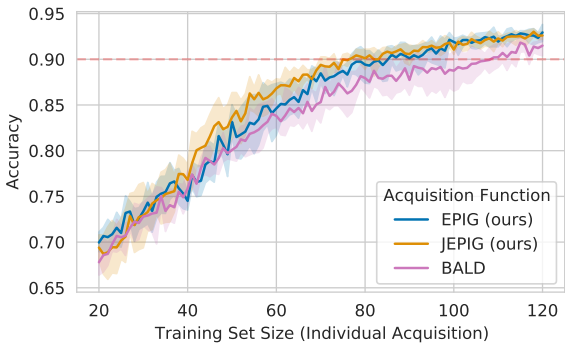


Figure 7.8: *EPIG vs JEPIG vs BALD with Bayesian Neural Networks on MNIST.* JEPIG performs better under than MC Dropout than EPIG.

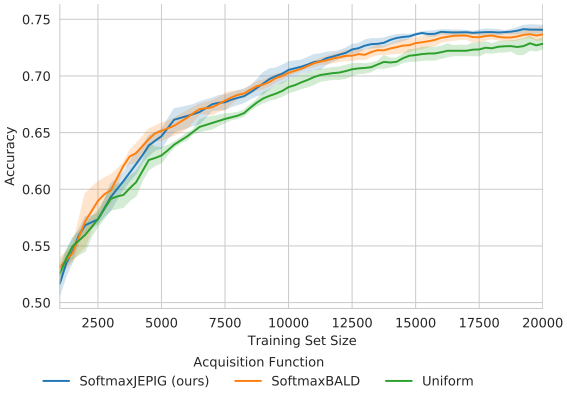


Figure 7.9: *BALD vs JEPIG on CIFAR-10.* JEPIG outperforms BALD. 5 trials each. With batch acquisition size 250, and initial training size 1000. Median accuracy after smoothing with a Parzen window filter over 30 acquisition steps to denoise.

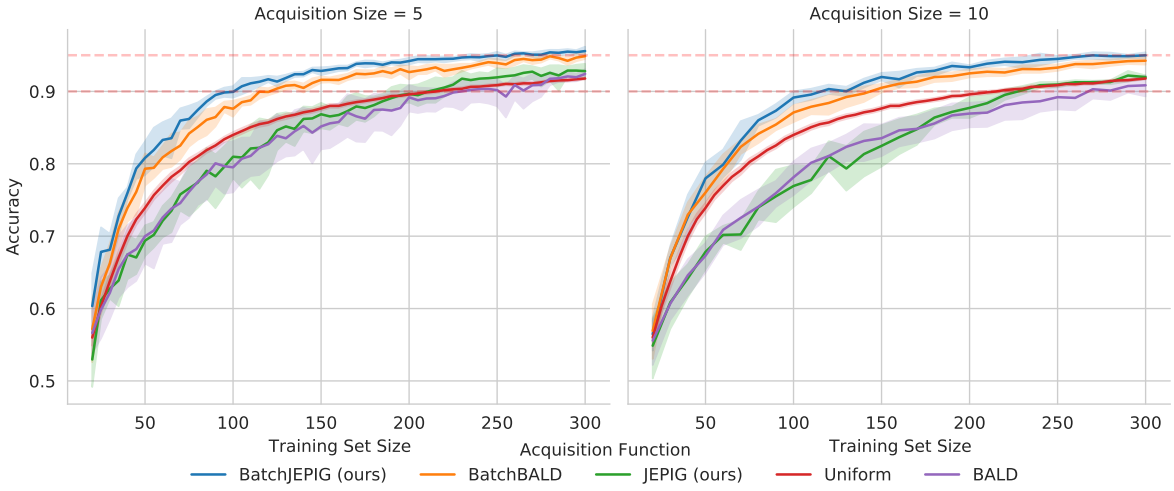
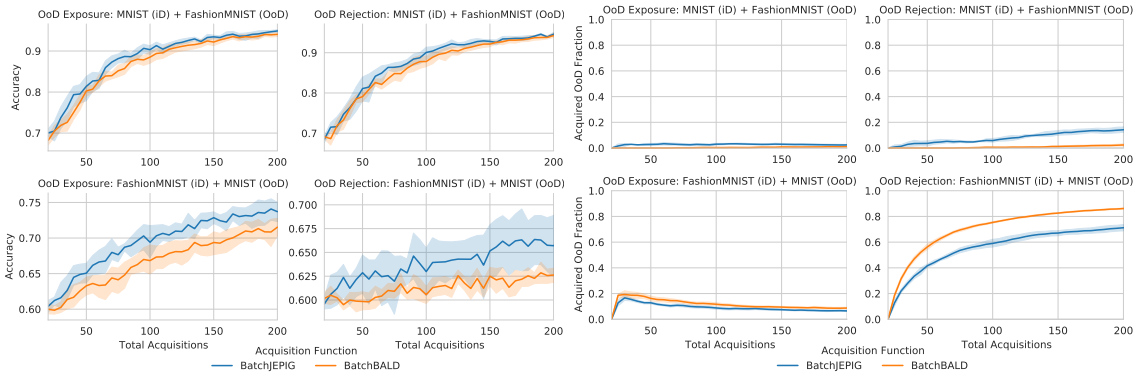


Figure 7.10: *(Batch)BALD vs (Batch)JEPIG on Repeated-MNIST (MNISTx2).* JEPIG outperforms BALD.

one or the other. We deal with an acquisition function attempting to acquire OoD data in two different modes: *OoD rejection* rejects OoD data from the batch and does not acquire it; while *OoD exposure* acquires OoD data with uniform targets, similar to outlier exposure methods in OoD detection [Hendrycks et al., 2019]. We use an evaluation set with 2000 unlabeled samples.

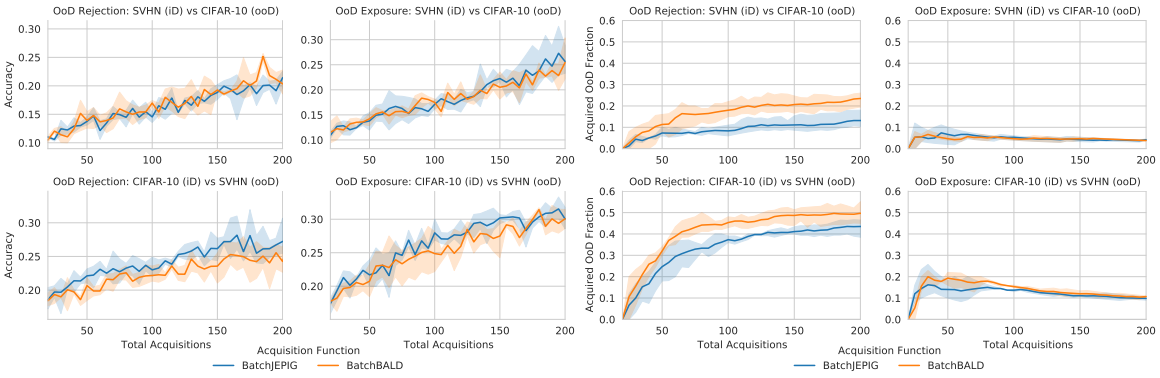
JEPIG outperforms BALD on in all combinations, see Figure 7.11(a). In all cases but one, JEPIG acquires fewer junk/OoD samples, see Figure 7.11(b). The ablation in Figure 7.13 shows that larger evaluation sets are beneficial. Note that the evaluation set is unlabeled and thus does not count towards sample acquisitions.

For CIFAR-10 and SVHN [Netzer et al., 2011], JEPIG outperforms BALD under distribution shift in all but one combination and selects fewer OoD samples, see Figure 7.12. We use an evaluation set with 1000 unlabeled samples.



(a) Accuracy **(b)** Acquired OoD Ratio

Figure 7.11: *MNIST and FashionMNIST pairings with OoD rejection or exposure.* JEPIG performs better than BALD. 5 trials.



(a) Accuracy **(b)** Acquired OoD Ratio

Figure 7.12: *CIFAR-10 and SVHN pairings with OoD rejection or exposure.* JEPIG performs better than BALD. 5 trials. Acquisition size 5. Initial training size 5.

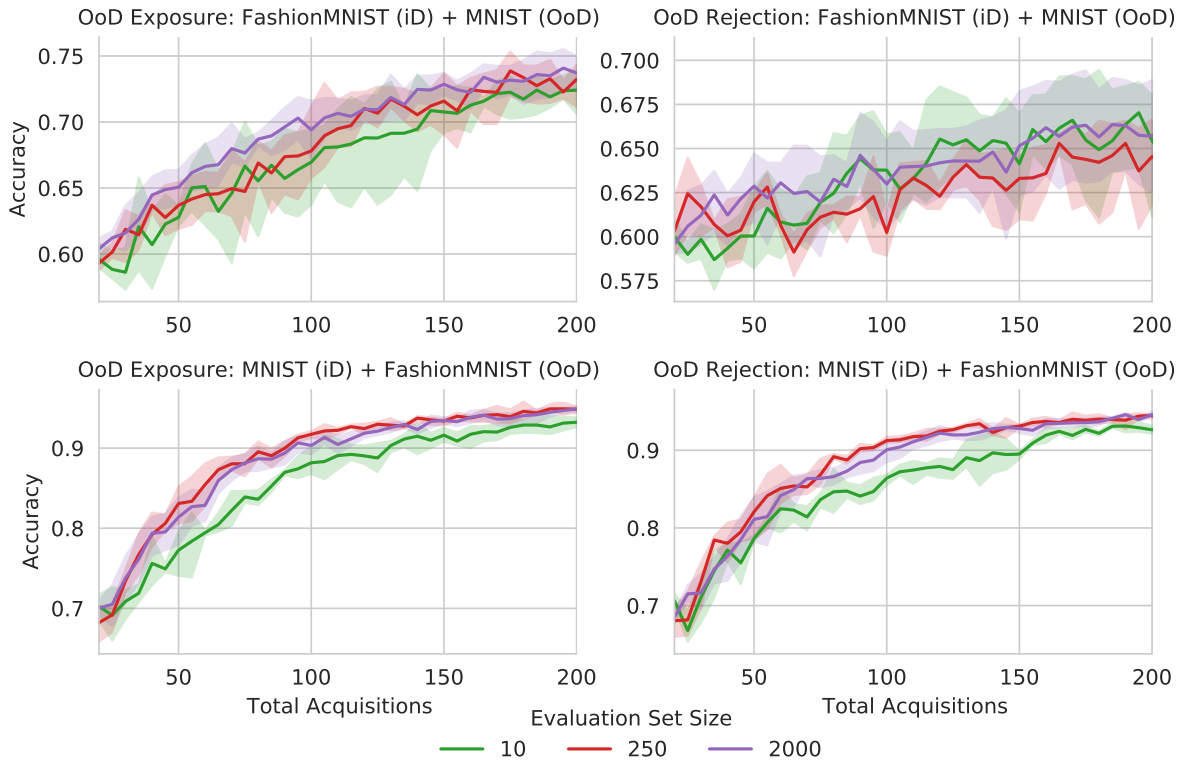


Figure 7.13: *Evaluation Set Size Ablation.* MNIST and FashionMNIST pairings with OoD rejection or exposure. A larger evaluation set performs better. 5 trials.

7.5 Related Work

The idea of using the EIG to quantify the utility of data was introduced by Lindley [1956] and has a long history of use in experimental design [Chaloner and Verdinelli, 1995a; Rainforth et al., 2023]. The framework of Bayesian experimental design has many applications outside active learning, and in these applications the model parameters are commonly the quantity of interest—Bayesian optimization [Hennig and Schuler, 2012; Hernández-Lobato et al., 2014; Villemonteix et al., 2009] being a notable exception. The EIG in the parameters is thus often a natural acquisition function.

The EIG in the parameters was originally suggested as an acquisition function for active learning by MacKay [1992a,b], who called it the total information gain. Despite its shortfalls, some of which were also briefly discussed by Freund et al. [1997] and MacKay [1992b] for special cases such as Bayesian linear regression, it has been widely used in cases where the model's predictions, not the model parameters, are the ultimate objects of interest [Atighehchian et al., 2020; Beluch et al., 2018; Gal et al., 2017; Houlsby et al., 2011; Jeon, 2020; Lee and Kim, 2019; Munjal et al., 2022; Pinsler et al., 2019; Shen et al., 2018; Siddhant and Lipton, 2018; Tran et al., 2019].

Maximizing the information gathered about a quantity other than the model parameters has been proposed a number of times as an approach to active learning. The predictive information as mutual information between the past and future was introduced by Bialek and Tishby [1999] and has been used to increase sample efficiency in reinforcement learning [Lee et al., 2020]. Perhaps most relevant to this chapter, MacKay [1992a,b] also discussed two other acquisition functions, namely the mean marginal information gain and the joint information gain, which correspond to EPIG and JEPIG, respectively.

The mean marginal information gain measures the average information gain in the predictions made on a fixed set of inputs based on a Gaussian approximation of the posterior over the model parameters. Though the mean marginal information gain has since received surprisingly little attention in the literature, it was discussed by Huszár [2013] and later used by Wang et al. [2021] to evaluate the quality of predictive-posterior correlations. Wang et al. [2021] extended the mean marginal information gain to the batch setting following insights from BatchBALD for regression tasks and evaluated it in a transductive active learning setting. Wang et al. [2021] only examine the case where pool and target input distribution are identical. The transductive approach to active learning [Vapnik, 2006; Yu et al., 2006] seeks to maximize the performance on a fixed set of inputs—in contrast with the input distribution considered by EPIG.

The joint information gain has received even less attention than the mean marginal information gain in the literature. This might be because MacKay showed that it is equivalent to BALD when assuming constant aleatoric noise (with a sufficiently large number of evaluation samples). Constant aleatoric noise is also a common and convenient choice for Gaussian Processes, which could explain why it was not considered by works in Bayesian optimization either. However, for deep neural networks, constant aleatoric noise is not a common assumption, and indeed the main benefit of using BALD over entropy is that it performs well when aleatoric uncertainty varies across samples because it estimates epistemic uncertainty and not aleatoric uncertainty (see also §3). Moreover, our detailed re-examination of the relationship between BALD and JEPIG reaches a more varied conclusion than MacKay [1992b] by specifically taking into account the support of pool and evaluation samples.

Aside from the work of MacKay [1992a,b], there are numerous prediction-oriented methods [Afrabandpey et al., 2019; Chapelle, 2005; Cohn, 1993; Cohn et al., 1996; Daee et al., 2017; Donmez and Carbonell, 2008; Evans et al., 2015; Filstroff et al., 2021; Krause et al., 2008; Seo et al., 2000; Sundin et al., 2018, 2019; Tan et al., 2021; Yu et al., 2006; Zhao et al., 2021a,b,c; Zhu et al., 2003]. Many of these, with notable examples including the work of Cohn et al. [1996] and Krause et al. [2008], are tied to a particular model class or approximation scheme and so lack EPIG's generality.

There is an additional limitation associated with techniques based on the idea, due to Roy and McCallum [2001], of measuring the expected loss reduction that would result from updating the model on a given input-label pair. These techniques often require updating the model within the computation of the acquisition function, which can be extremely expensive. Despite a strong conceptual connection to the acquisition function proposed by Roy and McCallum [2001], EPIG allows a significantly lower computational cost: its information-theoretic formulation allows us to derive an estimator that does not require nested model updating.

Unlike much active learning literature whose experiment setting implicitly assumes that that pool and test samples are drawn from the same distribution, EPIG and JEPIG support the setting in which pool and test distribution do *not* match. This is not the case for other diversity-based active learning methods such as BADGE [Ash et al., 2020], for example, which implicitly use the empirical pool set distribution to select diverse samples via clustering.

7.6 Discussion

We have demonstrated that BALD, a widely used acquisition function for Bayesian active learning, can be suboptimal. While much of machine learning focuses on prediction, BALD targets information gain in a model's parameters in isolation and so can seek labels that have limited relevance to the predictions of interest.

Motivated by this, we have proposed EPIG, an acquisition function that targets information gain in terms of predictions. Our results show EPIG outperforming BALD across a number of data settings (low- and high-dimensional inputs, varying degrees of divergence between the pool and target data distributions, and varying degrees of knowledge of the target distribution) and across multiple different models. This suggests EPIG can serve as a compelling drop-in replacement for BALD, with particular scope for performance gains when using large, diverse pools of unlabeled data.

While EPIG can be evaluated efficiently in the individual acquisition case, it becomes very costly for batch acquisition (when using the same estimators for classification). Our initial investigations of JEPIG with self-distillation shows it to be an approximation that is also well motivated and potentially faster to evaluate in the batch setting. However, it requires training an additional model. While this could be sped up by using warm-starting [Ash and Adams, 2020], it is still a significant cost. At the same time, two (Batch)BALD terms need to be computed for JEPIG which can also be slow and which does not scale well beyond small acquisition batch sizes. Thus, EPIG and JEPIG represent a trade-off between the time it takes to compute the expectation over evaluation samples and training another model. In the case of individual acquisition, EPIG is strongly favored. In §5, we discuss a simple stochastic extension to avoid computing BatchBALD terms. We do not examine this approach in detail for the acquisition functions presented in this chapter.

Importantly, unlike BALD, EPIG is not submodular, and thus greedy acquisition is not guaranteed to obtain $1 - \frac{1}{e}$ optimality for batch acquisition, even though we have not experienced any degradation in comparison with BALD/BatchBALD empirically in our initial experiments. Indeed, JEPIG performs on par or better than BALD in the regular active learning case without distribution shift. At the same time, while neither BALD nor JEPIG are adaptive submodular, and no statements about global optimality have been presented so far [Golovin and Krause, 2011; Foster, 2022], this has not been an issue in practice. Ash et al. [2021] recently introduced a new forward-backward strategy for a weight-space version of JEPIG. We leave an evaluation using the estimators we have used to future work.

Finally, we present two scenarios where EPIG might not perform better than BALD: Firstly, if a task's performance is dominated by a more general "sub-task", there might be many samples highly informative for this general sub-task. For image data, this could be the case when feature learning is of particular importance, and convolution kernels can be learned from image data, no matter the label or actual task. In this regime, both BALD and EPIG will perform similarly, yet will likely outperform random acquisition. Secondly, if the model's architecture and its inductive biases have specifically been evolved for a task (and dataset), there will be a high overlap between the model parameters and the task's target input distribution as model architectures which converge faster are preferred as research outputs. BALD and EPIG might perform similarly in this case, too. That is, prediction-oriented acquisition functions might show their strength when the task performance is highly specific and dependent on specific samples in the pool set, and the model's architecture is over-parameterized and not yet adapted to the task. We leave investigation of this for the future.

I am always in front of you but can never be seen.
What am I?

8

Prioritized Data Selection during Training

Now, let us take a look at active sampling. We will apply information-theoretic ideas to it. Active sampling is important because state-of-the-art models such as GPT-3 [Brown et al., 2020], CLIP [Radford et al., 2021], and ViT [Dosovitskiy et al., 2020] achieve remarkable results by training on vast amounts of web-scraped data, yet despite intense parallelization, training such a model takes weeks or months [Radford et al., 2021; Chowdhery et al., 2022]. Even practitioners who work with smaller models face slow development cycles, due to numerous iterations of algorithm design and hyperparameter selection. As a result, the total time required for training is a core constraint in the development of such deep learning models.

If it further sped up training, practitioners with sufficient resources would use much larger batches and distribute them across many more machines [Anil et al., 2018]. However, this has rapidly diminishing returns [LeCun et al., 2012], to a point where adding machines does not reduce training time [McCandlish et al., 2018; Anil et al., 2018]—see e.g. GPT-3 and PaLM [Chowdhery et al., 2022].

Additional machines can, however, still speed up training by filtering out less useful samples [Alain et al., 2015]. Many web-scraped samples are *noisy*, i.e. their label is incorrect or inherently ambiguous. For example, the text associated with a web-scraped image is rarely an accurate description of the image. Other samples are learned quickly and are then *redundant*. Redundant samples are commonly part of object classes that are over-represented in web-scraped data [Tian et al., 2021], and they can often be left out without losing performance. Given that web-scraped data is plentiful—often enough to finish training in a single epoch [Komatsuzaki, 2019; Brown et al., 2020]—one can afford to skip less useful points.

However, there is no consensus on which data points are the most useful. Some works, including curriculum learning, suggest prioritizing *easy* points with low label noise before training on all points equally [Bengio et al., 2009]. While this approach may improve convergence and generalization, it lacks a mechanism to skip points that are already learned (redundant). Other works instead suggest training on points that are *hard* for the model, thereby avoiding redundant points, whose loss cannot be further reduced. Online batch selection methods [Loshchilov and Hutter, 2015; Katharopoulos and Fleuret, 2018; Jiang et al., 2019] do so by selecting points with high loss or high gradient norm.

We show two failure modes of prioritizing hard examples. Firstly, in real-world noisy datasets, high loss examples may be mislabeled or ambiguous. Indeed, in controlled experiments, points selected by high loss or gradient norm are overwhelmingly those with noise-corrupted labels. Our results show that this failure mode degrades performance

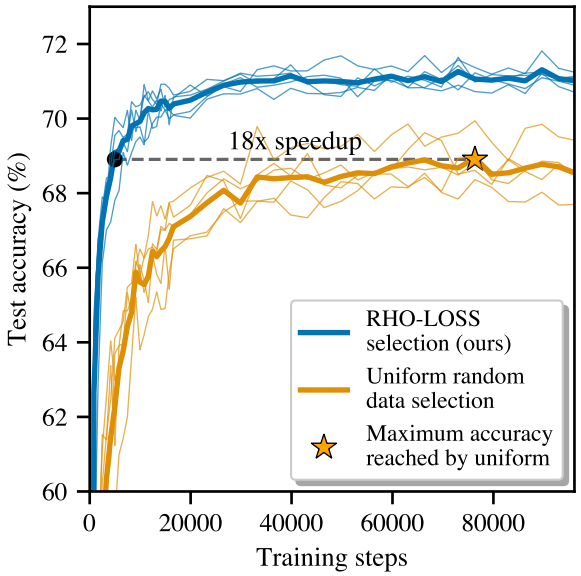


Figure 8.1: Speedup on large-scale classification of web-scraped data (Clothing-1M). RHO-LOSS trains all architectures with fewer gradient steps than standard uniform data selection (i.e. shuffling), helping reduce training time. Thin lines: ResNet-50, MobileNet v2, DenseNet121, Inception v3, GoogleNet, mean across seeds. Bold lines: mean across all architectures.

severely. More subtly, we show that some samples are hard because they are outliers—points with unusual features that are less likely to appear at test time. For the aim of reducing test loss, such points are less *worth learning*.

To overcome these limitations, we introduce the *reducible holdout loss selection* (RHO-LOSS) in this chapter. We propose a selection function grounded in probabilistic and information-theoretic modelling that quantifies by how much each point would reduce the loss on unseen data if we were to train on it, *without actually training on it*. We show that optimal points for reducing holdout loss are non-noisy, non-redundant, and task-relevant. To approximate optimal selection, we derive an efficient and easy-to-implement selection function: the reducible holdout loss.

We explore RHO-LOSS in extensive experiments on 7 datasets. We evaluate the reduction in required training steps compared to uniform sampling and state-of-the-art batch selection methods. Our evaluation includes Clothing-1M, the main large benchmark with noisy, web-scraped labels, matching our main application. RHO-LOSS reaches target accuracy in 18x fewer steps than uniform selection and achieves 2% higher final accuracy (Figure 8.1). Further, RHO-LOSS consistently outperforms prior art and speeds up training across datasets, modalities, architectures, and hyperparameter choices. Explaining this, we show that methods selecting "hard" points prioritize noisy and less relevant examples. In contrast, RHO-LOSS chooses low-noise, task-relevant, non-redundant points—points that are learnable, worth learning, and not yet learned.

8.1 Active Sampling: Online Batch Selection

Unlike the setting we use in most of the other chapters, which we introduced in §1.2.2, active sampling follows a different paradigm by virtue of being an online

algorithm and using labeled data. We will try to use most of the notation from §1.2.2 and §1.2.4; however, note that *the labels are available in the pool set and evaluation set in this chapter.*

We consider a model $\mathrm{p}(y \mid \boldsymbol{x}, \theta)$ with parameters θ that we want to train using stochastic gradient descent (SGD) on a subset of samples from the available *labeled* data $\mathcal{D}^{\text{pool}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{\mathtt{M}}$ using the cross-entropy loss. At each training step t, we load a batch b_t of size $\mathtt{K}$ from $\mathcal{D}^{\text{pool}}$. In online batch selection [Loshchilov and Hutter, 2015], we uniformly pre-sample a larger batch B_t of size $\mathtt{K}' \gg \mathtt{K}$. Then, we construct a smaller batch b_t that consists of the top-ranking $\mathtt{K}$ points in B_t ranked by a label-aware selection function $S(\boldsymbol{x}_i, y_i)$. We perform a gradient step to minimize a mini-batch loss $\mathrm{H}[y_i \mid \boldsymbol{x}_i, \theta]$ summed over $i \in b_t$. The next large batch B_{t+1} is then pre-sampled from $\mathcal{D}^{\text{pool}}$ without replacement of previously sampled points (points are only replaced at the start of the next epoch).

8.2 (Joint) Predictive Information Gain & Reducible Holdout Loss Selection

Previous online batch selection methods, such as loss or gradient norm selection, aim to select points that, if we were to train on them, would minimize the *training set* loss. [Loshchilov and Hutter, 2015; Katharopoulos and Fleuret, 2018; Kawaguchi and Lu, 2020; Alain et al., 2015]. Instead, we aim to select points that minimize the loss on a *holdout set*. In spirit of this thesis, we will refer to this holdout set as a *labeled* evaluation set[1] $\mathcal{D}^{\text{eval}}$. It would be too expensive to naively train on every candidate point and evaluate the holdout loss each time.

In this section, we show how to (approximately) find the points that would most reduce the holdout loss if we were to train the current model on them (without actually training on them). For simplicity, we first assume only one point $(\boldsymbol{x}, y) \in B_t$ is selected for training at each time step t (we discuss selection of multiple points below). $\mathrm{p}(y' \mid \boldsymbol{x}'; \mathcal{D}^{\text{train}})$ is the predictive distribution of the current model, where $\mathcal{D}^{\text{train}}$ is the sequence of data the model was trained on before training step t. $\mathcal{D}^{\text{eval}} = \{(\boldsymbol{x}_i^{\text{eval}}, y_i^{\text{eval}})\}_{i=1}^{\mathtt{E}}$, written as $\boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}$ and $y_{1..\mathtt{E}}^{\text{eval}}$, respectively, for brevity, is a *labeled* evaluation set drawn from the same data-generating distribution $\mathrm{p}_{\text{true}}(\boldsymbol{x}', y')$ as is the set of available training data $\mathcal{D}^{\text{pool}}$—this is the holdout set.

Reduction in Holdout Loss. We aim to acquire the point $(\boldsymbol{x}, y) \in B_t$ that, if we were to train on it, would minimize the cross-entropy loss on the holdout set, our *labeled* evaluation set:

$$\operatorname*{arg\,min}_{(x,y)\in B_t} \mathrm{H}(\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}}, y^{\text{eval}}) \parallel \mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}; (y, \boldsymbol{x}), \mathcal{D}^{\text{train}})). \tag{8.1}$$

Thinking back to §7, we see that in the active learning case, we were looking for a reduction in expected loss, while here we are looking for a reduction in the holdout loss.

(Joint) Predictive Information Gain. Using the notation from §2, the cross-entropy loss over the empirical $\mathcal{D}^{\text{eval}}$ is just:

$$\mathrm{H}(\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}}, y^{\text{eval}}) \parallel \mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, (y, \boldsymbol{x}), \mathcal{D}^{\text{train}})) \tag{8.2}$$

[1] Supposedly, we should then also rename the loss to REV-LOSS in this thesis, but let us not.

$$= \mathbb{E}_{\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}}, y^{\mathrm{eval}})}[\mathrm{H}[y^{\mathrm{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, (y, \boldsymbol{x}), \mathcal{D}^{\mathrm{train}}]]. \tag{8.3}$$

Thus, we can write the minimization above as a maximization of a (point-wise) mutual information analogously to derivations in §7:

$$\arg\min \mathbb{E}_{\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}}, y^{\mathrm{eval}})}[\mathrm{H}[y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, (y, \boldsymbol{x}), \mathcal{D}^{\mathrm{train}}]] \tag{8.4}$$

$$= \arg\max \mathbb{E}_{\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}}, y^{\mathrm{eval}})}[\mathrm{I}[y^{\mathrm{eval}}; y \mid \boldsymbol{x}^{\mathrm{eval}}, \boldsymbol{x}, \mathcal{D}^{\mathrm{train}}]]. \tag{8.5}$$

This is the predictive information gain:

Definition 8.1. The *predictive information gain* (PIG) of a point $(\boldsymbol{x}, y)$ is defined as:

$$\mathrm{PIG}(\boldsymbol{x}, y) = \mathbb{E}_{\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}}, y^{\mathrm{eval}})}[\mathrm{I}[y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, (y, \boldsymbol{x}), \mathcal{D}^{\mathrm{train}}]]. \tag{8.6}$$

The motivation for it is the same as for EPIG in §7.3 with the crucial different label information is available here, making batch evaluation easier.

Similarly, we can define a joint predictive information gain.

Definition 8.2. The *joint predictive information gain* (JPIG) of a point $(\boldsymbol{x}, y)$ is defined as:

$$\mathrm{JPIG}(\boldsymbol{x}, y) = \mathrm{I}[y^{\mathrm{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, (y, \boldsymbol{x}), \mathcal{D}^{\mathrm{train}}] \tag{8.7}$$

$$= \mathrm{H}[y^{\mathrm{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}} \mathcal{D}^{\mathrm{train}}] - \mathrm{H}[y^{\mathrm{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, (y, \boldsymbol{x}), \mathcal{D}^{\mathrm{train}}] \tag{8.8}$$

$$= -\log \mathrm{p}(y^{\mathrm{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\mathrm{train}}) - (-\log \mathrm{p}(y^{\mathrm{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\mathrm{eval}}_{1..\mathtt{E}}, (y, \boldsymbol{x}), \mathcal{D}^{\mathrm{train}})). \tag{8.9}$$

The motivation for JPIG is also the same as for JEPIG in §7.4.2, again with the crucial difference that label information is available here. JPIG is even closer to the minimization of the log marginal likelihood, which is usually the proxy goal of Bayesian model selection.

In §6, we will take a different perspective on these quantities and examine them as marginal and joint cross-entropies (or marginal and joint cross-mutual information), which is more sensible, perhaps. However, we will still use the notation of PIG and JPIG for the rest of this chapter and this thesis otherwise, as it ties the different information quantities together more nicely[2].

Deriving a Tractable Selection Function. We now derive a tractable expression for the term in Eq. (8.1) that does not require us to train on each candidate point $(\boldsymbol{x}, y) \in B_t$ and then evaluate the loss on $\mathcal{D}^{\mathrm{eval}}$. To make our claims precise and our assumptions transparent, we use the language of Bayesian probability theory. We treat model parameters as a random variable with prior $\mathrm{p}(\theta)$ and infer a posterior $\mathrm{p}(\theta \mid \mathcal{D}^{\mathrm{train}})$ using the already-seen training data $\mathcal{D}^{\mathrm{train}}$. The model has a predictive distribution $\mathrm{p}(y \mid \boldsymbol{x}, \mathcal{D}^{\mathrm{train}}) = \int_\theta \mathrm{p}(y \mid \boldsymbol{x}, \theta)\, \mathrm{p}(\theta \mid \mathcal{D}^{\mathrm{train}}) d\theta$. When using a point estimate of θ, the predictive distribution can be written as an integral with respect to a Dirac delta.

We have already motivated that Equation 8.1 is equivalent to maximizing the PIG objective. As a first step, we will switch to using the JPIG objective (**Approximation 0**), which is more closely related to the log marginal likelihood and model selection.

[2] Although we could also follow a different naming scheme and call them T(E)IG, J(E)IG, and M(E)IG, which would be more similar to MacKay [1992b]. Naming things is not easy.

We will show that it leads to a tractable selection function: We simply rewrite JPIG using the symmetry of the point-wise mutual information as:

$$\mathrm{I}[y^{\text{eval}}_{1..\mathtt{E}} \mid \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, (y, \boldsymbol{x}), \mathcal{D}^{\text{train}}] = \mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] - \mathrm{H}[y \mid \boldsymbol{x}, y^{\text{eval}}_{1..\mathtt{E}}, \boldsymbol{x}^{\text{eval}}_{1..\mathtt{E}}, \mathcal{D}^{\text{train}}]. \tag{8.10}$$

As exact Bayesian inference (conditioning on $\mathcal{D}^{\text{train}}$ or $\mathcal{D}^{\text{eval}}$) is intractable in neural networks [Blundell et al., 2015], we fit the models with SGD instead (**Approximation 1**). We study the impact of this approximation in §8.3.1. The first term, $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$, is then the *training loss* on the point $(\boldsymbol{x}, y)$ using the current model trained on $\mathcal{D}^{\text{train}}$. The second term, $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}, \mathcal{D}^{\text{train}}]$, is the loss of a model trained on $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{eval}}$.

Although the selection function in Eq. (8.10) is tractable, it is still somewhat expensive to compute, as both terms must be updated after each acquisition of a new point. However, we can approximate the second term with a model trained only on the holdout dataset, $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}, \mathcal{D}^{\text{train}}] \approx \mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}]$ (**Approximation 2**). This approximation saves a lot of compute: it is now sufficient to compute the term once before the first epoch of training. Later on, we show that this approximation empirically does not hurt performance on any tested dataset and even has additional benefits (§8.3.1 and Appendix G.4). We term $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}]$ the *irreducible holdout loss* (IL) since it is the remaining loss on point $(\boldsymbol{x}, y) \in \mathcal{D}^{\text{pool}}$ after training on the holdout set (*labeled* evaluation set) $\mathcal{D}^{\text{eval}}$; in the limit of $\mathcal{D}^{\text{eval}}$ being large, it would be the lowest loss that the model can achieve without training on $(\boldsymbol{x}, y)$. Accordingly, we name our approximation of Eq. (8.10) the *reducible holdout loss*—the difference between the training loss and the irreducible holdout loss (IL).

Our method still requires us to train a model on a holdout set (*labeled* evaluation set), but a final approximation greatly reduces that cost. We can efficiently compute the IL with an "irreducible loss model" (IL model) that is smaller than the target model and has low accuracy (**Approximation 3**). We show this and explain it in Sections 8.3.1, 8.3.2, and 8.3.3. Counterintuitively, the reducible holdout loss can therefore be negative. Additionally, one IL model can be reused for many target model runs, amortizing its cost (§8.3.2). For example, we trained all 40 seeds of 5 target architectures in Figure 8.1 using a single ResNet18 IL model. Further, this model trained for 37x fewer steps than each target model (reaching only 62% accuracy). §8.4 details further possible efficiency improvements.

In summary, selecting a point that minimizes the holdout loss in Eq. (8.1), for a model trained on $\mathcal{D}^{\text{train}}$, can be approximated with the following easy-to-compute objective:

Reducible holdout loss selection (RHO-LOSS)

$$\underset{(\boldsymbol{x},y)\in B_t}{\arg\max} \overbrace{\underbrace{\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]}_{\text{training loss}} - \underbrace{\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}]}_{\text{irreducible holdout loss (IL)}}}^{\text{reducible holdout loss}} \tag{8.11}$$

Although we required additional data $\mathcal{D}^{\text{eval}}$, this is not essential for large (§8.3.0) nor small (§8.3.2) datasets.

Understanding the Reducible Loss. We now provide intuition on why reducible holdout loss selection (RHO-LOSS) avoids redundant, noisy, and less relevant points.
i) Redundant points. We call a training point redundant when the model has already

Algorithm 3 Reducible holdout loss selection (RHO-LOSS)

Input: Small model $p(y \mid \boldsymbol{x}; \mathcal{D}^{\text{eval}})$ trained on a *labeled evaluation set* $\mathcal{D}^{\text{eval}}$ (or holdout set), batch size $\texttt{K}$, large batch size $\texttt{K}' > \texttt{K}$, learning rate η.

for $(\boldsymbol{x}_i, y_i)$ in `training set` **do**
    `IrreducibleLoss[i]` $\leftarrow \mathrm{H}[y_i \mid \boldsymbol{x}_i, \mathcal{D}^{\text{eval}}]$

Initialize parameters θ^0 and set $t = 0$

for $t = 0, 1, \ldots$ **do**
    Randomly select a large batch B_t of size $\texttt{K}'$.
    $\forall i \in B_t$, compute `Loss[i]`, the train loss of point i given parameters θ^t
    $\forall i \in B_t$, compute `RHOLOSS[i]` $\leftarrow$ `Loss[i]` $-$ `IrreducibleLoss[i]`
    $b_t \leftarrow$ top-$\texttt{K}$ samples in B_t in terms of `RHOLOSS`.
    $g_t \leftarrow$ mini-batch gradient on b_t using parameters θ^t
    $\theta^{t+1} \leftarrow \theta^t - \eta g_t$

learned it, i.e. its training loss cannot be further reduced. Since redundant points have low training loss, and the reducible loss is always less than the training loss (Eq. (8.11)), such points have low reducible loss and are not selected. And if the model forgets them, they are revisited in the next epoch. **ii) Noisy points.** While prior methods select based on high training loss (or gradient norm), not all points with high loss are informative—some may have an ambiguous or incorrect (i.e. noisy) label. The labels of such points cannot be predicted using the holdout set (*labeled* evaluation set) [Chen et al., 2019]. Such points have high IL and, consequently, low reducible loss. These noisy points are less likely to be selected compared to equivalent points with less noise. **iii) Less relevant points.** Loss-based selection has an additional pitfall. The training loss is likely higher for outliers in input space—values of $\boldsymbol{x}$ far from most of the training data, in regions with low input density under $p_{\text{true}}(\boldsymbol{x})$. Points with low $p_{\text{true}}(\boldsymbol{x})$ should not be prioritized, all else equal. Consider an 'outlier' $(\boldsymbol{x}, y)$ and a non-outlier $(\boldsymbol{x}', y')$, with $p_{\text{true}}(\boldsymbol{x}) < p_{\text{true}}(\boldsymbol{x}')$ but *equal* training loss $\mathrm{H}[y|\boldsymbol{x}, \mathcal{D}^{\text{train}}] = \mathrm{H}[y'|\boldsymbol{x}', \mathcal{D}^{\text{train}}]$. As the holdout set (*labeled* evaluation set) $\mathcal{D}^{\text{eval}}$ is also drawn from p_{true}, $\mathcal{D}^{\text{eval}}$ will contain fewer points from the region around $\boldsymbol{x}$ in input space compared to the region around $\boldsymbol{x}'$. Thus, training on $(\boldsymbol{x}, y)$ is likely to reduce the holdout loss (Eq. (8.1)) less, and so we prefer to train on the non-outlier $(\boldsymbol{x}', y')$. In the specific sense described, $(\boldsymbol{x}, y)$ is thus less *relevant* to the holdout set (*labeled* evaluation set). As desired, RHO-LOSS deprioritizes $(\boldsymbol{x}, y)$: since $\mathcal{D}^{\text{eval}}$ contains few points from the region around $\boldsymbol{x}$, the IL of $(\boldsymbol{x}, y)$ will be large.

In short, RHO-LOSS *deprioritizes* points that are redundant (low training loss), noisy (high IL), or less relevant to the holdout set (high IL). That is, RHO-LOSS *prioritizes* points that are not yet learned, learnable, and worth learning. We provide empirical evidence for these claims in §8.3.3. See Algorithm 3 for the implementation of RHO-LOSS.

Batch Selection. We showed which point is optimal when selecting a single point $(\boldsymbol{x}, y)$. When selecting an entire batch b_t, we select the points with the top-$\texttt{K}$ scores from the randomly pre-sampled set B_t. This is nearly optimal when assuming that each point has little effect on the score of other points, which is often used as a simplifying assumption in active learning. This assumption is much more reasonable

in our case than in active learning because model predictions are not changed much by a single gradient step on one mini-batch.

Simple Batch Selection. For large-scale neural network training, practitioners with sufficient resources would use many more machines if it further sped up training [Anil et al., 2018]. However, as more workers are added in synchronous or asynchronous gradient descent, the returns diminish to a point where adding more workers does not further improve wall clock time [Anil et al., 2018; McCandlish et al., 2018]. For example, there are rapidly diminishing returns for using larger batch sizes or distributing a given batch across more workers, for multiple reasons [McCandlish et al., 2018; Keskar et al., 2017]. The same holds for distributing the model across more workers along its width or depth dimension [Rasley et al., 2020; Shoeybi et al., 2019; Huang et al., 2019]. However, we can circumvent these diminishing returns by adding a new dimension of parallelization, namely, for data selection.

Since parallel *forward* passes do not suffer from such diminishing returns, one can use extra workers to evaluate training losses in parallel [Alain et al., 2015]. The theoretical runtime speedup can be understood as follows. The cost per training step of computing the selection function on B_t is $\frac{K'}{3K}$ times as much as the cost of the forward-backward pass needed to train on b_t since a forward pass requires at least 3x less computation than a forward-backward pass [Jouppi et al., 2017]. One can reduce the time for the selection phase almost arbitrarily by adding more workers that compute training losses using a copy of the model being trained. The limit is reached when the time for selection is dominated by the communication of parameter updates to workers. More sophisticated parallelization strategies allow reducing the time overhead even further (§8.4). To avoid assumptions about the particular strategy used, we report experiment results in terms of the required number of training epochs.

8.3 Empirical Validation

We evaluate our selection method on several datasets (both in controlled environments and real-world conditions) and show significant speedups compared to prior art, in the process shedding light on the properties of different selection functions.

Recall that our setting assumes training time is a bottleneck, but data is abundant—more than we can train on (see Bottou and LeCun [2003]). This is common e.g. for web-scraped data where state-of-the-art performance is often reached in less than half of one epoch [Komatsuzaki, 2019; Brown et al., 2020]. As data is abundant, we can set aside a holdout set (*labeled* evaluation set) for training the IL model with little to no downside. For the large Clothing-1M dataset, we implement RHO-LOSS by training the IL model on 10% of the training data, while all baselines are trained on the full 100% of the training data. For the smaller datasets, we simulate abundance of data by reserving a holdout set and training *all* methods only on the remaining data. However, RHO-LOSS also works on small datasets without additional data by double-using the training set (§8.3.2).

Datasets. We evaluate on 7 datasets: 1) QMNIST [Yadav and Bottou, 2019] extends MNIST [LeCun et al., 1998] with 50k extra images which we use as the holdout set (*labeled* evaluation set). 2) On CIFAR-10 [Krizhevsky, 2009] we train on half of the training set and use the other half as a holdout to train the irreducible loss (IL) model. 3) CIFAR-100: same as CIFAR-10. 4) CINIC-10 [Darlow et al., 2018] has 4.5x more

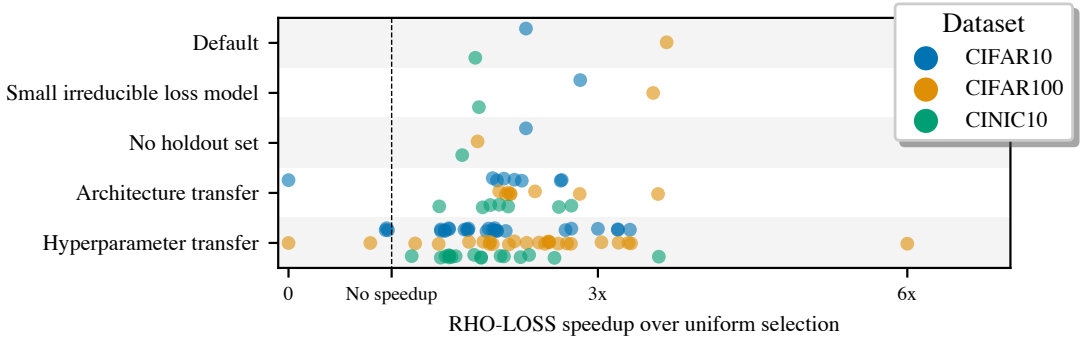


Figure 8.2: The irreducible loss model can be small, trained with no holdout data, and reused across target architectures and hyperparameters. Here, we use clean datasets, where speedups are smallest. The x-axis shows speedup, i.e. after how many fewer epochs RHO-LOSS exceeds the highest accuracy uniform selection achieves within 100 epochs. Row 1 uses a ResNet18 as irreducible loss model. All other rows instead use a small, cheap CNN. Each dot shows an experiment with a given combination of irreducible loss model and target model (mean across 2-3 seeds for all but the last row).

images than CIFAR-100 and includes a validation set (which we use as holdout set) and a test set with 90k images each. 5) Clothing-1M [Xiao et al., 2015], which contains over 1 million 256x256-resolution clothing images from 14 classes. The dataset is fully web-scraped—a key application area of this chapter—and is the most widely accepted benchmark for image recognition with noisy labels [Algan and Ulusoy, 2021]. We use the whole training set for training and reuse 10% of it to train the IL model. We further evaluate on two NLP datasets from GLUE [Wang et al., 2019]: 6) CoLA (grammatical acceptability) and 7) SST-2 (sentiment). We split their training sets as for CIFAR.

Baselines. Aside from uniform sampling (without replacement, i.e. random shuffling), we also compare to selection functions that have achieved competitive performance in online batch selection recently: the (training) loss, as implemented by Kawaguchi and Lu [2020], gradient norm, and gradient norm with importance sampling (called *gradient norm IS* in our figures), as implemented by Katharopoulos and Fleuret [2018]. We also compare to the core-set method Selection-via-Proxy (SVP) that selects data offline before training [Coleman et al., 2020]. We report results using maximum entropy SVP and select with the best-performing model, ResNet18. We further compare to four baselines from active learning, shown in Appendix G.7 as they assume labels are unobserved. Finally, we include selection using the negative IL (see Eq. 8.11) to test if it is sufficient to only skip noisy and less relevant but not redundant points.

Models and Hyperparameters. To show our method needs no tuning, we use the PyTorch default hyperparameters (with the AdamW optimizer [Loshchilov and Hutter, 2019]) and $\frac{K}{K'} = 0.1$. We test many additional hyperparameter settings in Figs. 8.2 (row 5) and G.5. We test various architectures in Figs. 8.1 and 8.2 (row 4). In all other figures, we use a 3 layer MLP for experiments on QMNIST, a ResNet-18 adapted for small images for CIFAR-10/CIFAR-100/CINIC-10, and a ResNet-50 for Clothing-1M. All models for Clothing-1M are pre-trained on ImageNet (standard for this dataset Algan and Ulusoy [2021]) and the IL model is *always* a ResNet-18. For the NLP datasets, we use a pre-trained ALBERT v2 [Lan et al., 2020]. We always use

Table 8.1: Spearman's rank correlation of rankings of data points by selection functions that are increasingly less faithful approximations of Eq. (8.10), compared to the most faithful approximation. Approximations added from left to right. Mean across 3 seeds.

	Non-Bayesian	Not converged	Not updating IL model	Small IL model
Rank correlation	0.75	0.76	0.63	0.51

the IL model checkpoint with the lowest validation loss (not the highest accuracy); this performs best. Details in Appendix G.2.

Evaluation. We measure speedup in terms of the number of epochs needed to reach a given test accuracy. We measure epochs needed, rather than wall clock time, as our focus is on evaluating a new selection function, not an entire training pipeline. Wall clock time depends primarily on the hardware used and implementation details that are beyond our scope. Most importantly, data selection is amenable to parallelization beyond standard data parallelism as discussed in §8.2.

8.3.1 Impact of Approximations

In §8.2, we introduced a function for selecting exactly the points that most reduce the model's loss on a holdout set (*labeled* evaluation set). To make this selection function efficient for deep neural networks, we made several approximations. Here, we study how these approximations affect the points selected, by successively introducing one approximation after the other.

Because the exact selection function (Eq. (8.10)) is intractable, we start with a close (and expensive) approximation as the gold standard (Approximation 0). To make Approximation 0 feasible, the experiments are conducted on an easy dataset—QMNIST (with 10% uniform label noise and data duplication to mimic the properties of web-scraped data). We then successively introduce the Approximations 1, 2, and 3 described in §8.2. To assess the impact of each approximation, we train a model without and with the approximations, and then compute the rank correlation (Spearman's correlation coefficient) of the selection function evaluated on each batch B_t. Across the first epoch, we present the mean of the rank correlations. Since each approximation selects different data, the corresponding models become more different over time; this divergence likely causes some of the observed differences in the points they select. See Appendix G.5 for details.

Approximation 0. To get as close as possible to the Bayesian inference/conditioning used in Eq. (8.10), we use a deep ensemble of 5 neural networks and train them *to convergence* after every time step t on the acquired dataset $b_t \cup \mathcal{D}^{\text{train}}$ Wilson and Izmailov [2020].

Approximation 1: SGD instead of Bayesian inference/conditioning. Approximation 0 is a close approximation of Eq. (8.10), but training an ensemble to convergence at every step t is far too expensive in practice. Starting from this gold-standard, we introduce two stronger approximations (1a and 1b) to move to standard neural network fitting with AdamW. 1a) First, we replace the ensemble with a single model, while still training to convergence at each time step. The Spearman's coefficient between

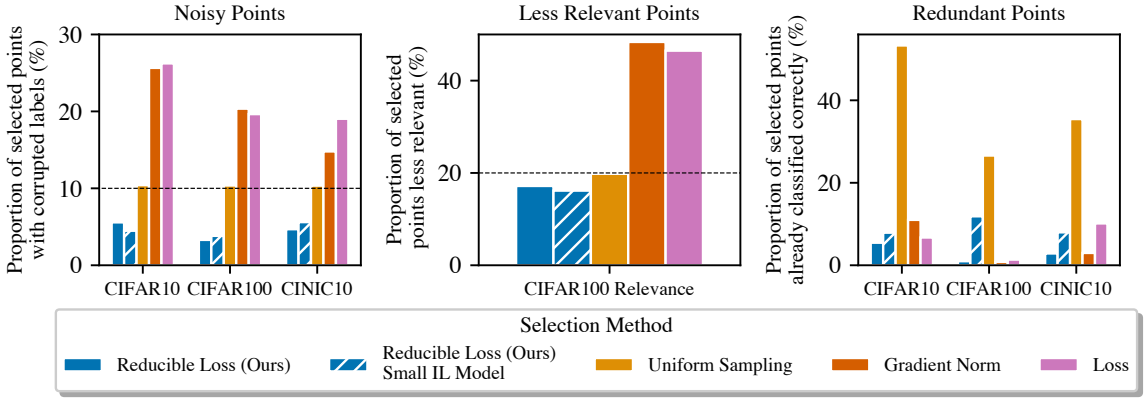


Figure 8.3: Properties of RHO-LOSS and other methods. RHO-LOSS prioritizes points that are non-noisy, task-relevant, and non-redundant—even when the irreducible loss (IL) model is a small CNN. In contrast, loss and gradient norm prioritize noisy and less relevant points (while also avoiding redundant points). **Left.** Proportion of selected points with corrupted labels. We added 10% uniform label noise, i.e., we randomly switched each point's label with 10% probability. **Middle.** Proportion of selected points from low relevance classes on CIFAR100 Relevance dataset. **Right.** Proportion of selected points that are already classified correctly, which is a proxy for redundancy. Mean over 150 epochs of training and 2-3 seeds.

this approximation and Approximation 0 is 0.75, suggesting similar points are selected ("Non-Bayesian" in Table 8.1). 1b) Next, we only take one gradient step on each new batch b_t. The Spearman's coefficient, when comparing this to Approximation 0, is 0.76 ("Not Converged" in Table 8.1).

Approximation 2. Not updating the IL model on the acquired data $\mathcal{D}^{\text{train}}$. Second, we save compute by approximating $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}, \mathcal{D}^{\text{eval}}]$ with $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}]$. The points selected are still similar to Approximation 0 (Spearman's coefficient 0.63, "Not updating IL model" in Table 8.1). This approximation also performs well on other datasets (Appendix G.4).

Approximation 3: Small IL model. Lastly, we use a model with 256 hidden units instead of 512 (4x fewer parameters) as the IL model and see again that similar points are selected (Spearman's coefficient 0.51). We study cheaper IL models in other forms and datasets in the next section.

8.3.2 Cheap Irreducible Loss Models & Robustness

RHO-LOSS requires training an IL model on a holdout set (*labeled* evaluation set), which poses additional costs. Here, we show how to minimize these costs and amortize them across many training runs of target models. The same experiments also show the robustness of RHO-LOSS across architectures and hyperparameter settings. To fit our computational budget, we perform these experiments on moderate-sized clean benchmark datasets although RHO-LOSS speeds up training more on noisy or redundant web-scraped data (see §8.3.4).

Irreducible Loss Models: Small & Cheap. In our default setting (Figure 8.2, row 1), both the target model and IL model have the same architecture (ResNet-18). In rows 2 and below, we instead used a small CNN similar to LeNet as the IL model

[LeCun et al., 1989]. It has 21x fewer parameters and requires 29x fewer FLOP per forward pass than the ResNet-18. **The smaller IL model accelerates training as much or more than the larger model**, even though its final accuracy is far lower than the target ResNet-18 (11.5% lower on CIFAR-10, 7% on CIFAR-100, and 8.1% on CINIC-10). We examine in §8.3.3 why this useful result holds.

Irreducible Loss Models: No Holdout Data. Web-scraped datasets are often so large that even a small fraction of the overall data can be sufficient to train the IL model. E.g., in our experiments on Clothing-1M (Figure 8.1), the holdout set (*labeled* evaluation set) is only 10% as large as the main train set. Additionally, we can train the IL model without any holdout data (Figure 8.2, row 3). We split the training set $\mathcal{D}^{\text{pool}}$ into two halves and train an IL model on each half (still using small IL models). Each model computes the IL for the half of $\mathcal{D}^{\text{pool}}$ that it was not trained on. Training two IL models costs no additional compute since each model is trained on half as much data compared to the default settings.

Irreducible Loss Models: Reuse for Different Target Architectures. We find that a single small CNN IL model accelerates the training of 7 target architectures (Figure 8.2, row 4): VGG11 (with batch norm), GoogleNet, Resnet34, Resnet50, Densenet121, MobileNet-v2, Inception-v3. RHO-LOSS does not accelerate training on CIFAR-10 for VGG11, which is also the architecture on which uniform training performs the worst; i.e. RHO-LOSS empirically does not "miss" a good architecture. Not only is RHO-LOSS robust to architectures choice, a single IL model can also be *reused by many practitioners* who use different architectures (as we did in Figure 8.1).

Irreducible Loss Models: Reuse for Hyperparameter Sweeps. We find that a single small CNN accelerates the training of ResNet-18 target models across a hyperparameter grid search (Figure 8.2, last row). We vary the batch size (160, 320, 960), learning rate (0.0001, 0.001, 0.01), and weight decay coefficient (0.001, 0.01, 0.1). RHO-LOSS speeds up training compared to uniform on nearly all target hyperparameters. The few settings in which it doesn't speed up training are also settings in which uniform training performs very poorly ($< 30\%$ accuracy on CIFAR-100, $< 80\%$ on CIFAR-10).

8.3.3 Properties of RHO-LOSS & Other Selection Functions

We established that RHO-LOSS can accelerate the training of various target architectures with a single IL model, even if the IL model is smaller and has considerably lower accuracy than the target models (§8.3.2). This suggests robustness to target-IL architecture mismatches.

To understand this robustness, we investigate the properties of points selected by RHO-LOSS, when the target and IL model architectures are identical, and when they differ. In both cases, we find that RHO-LOSS prioritizes points that are non-noisy, task-relevant, and not redundant. We also investigate the properties of points selected by prior art.

Noisy Points. We investigate how often different methods select noisy points by uniformly corrupting the labels for 10% of points and tracking what proportion of selected points are corrupted. RHO-LOSS deprioritizes noisy points for both IL models (Figure 8.3). We observe a failure mode of the widely-used loss and gradient norm selection functions: they select far more noisy points than uniform. These methods also

Table 8.2: Epochs required to reach a given target test accuracy (final accuracy in parentheses). Figs. G.1 and G.2 (Appendix) show all training curves. Some datasets have 10% uniform label noise added. Results averaged across 2-4 seeds. Best performance in **bold**. RHO-LOSS performs best in both epochs required and final accuracy. *NR* indicates that the target accuracy was not reached. *On CIFAR10/100, CoLA, and SST-2, only half of the data is used for training (§8.3.0).

Dataset	Target Acc	Number of epochs method needs to reach target accuracy ↓ (Final accuracy in parentheses)						
		Train Loss	Grad Norm	Grad Norm IS	SVP	Irred Loss	Uniform	RHO-LOSS
Clothing-1M	60.0%	8	13	2	*NR*	*NR*	2	**1**
	69.0%	*NR* (65%)	*NR* (64%)	9 (70%)	*NR* (55%)	*NR* (48%)	30 (70%)	**2 (72%)**
CIFAR10*	80.0%	81	*NR*	57	*NR*	*NR*	79	**39**
	87.5%	129 (90%)	*NR* (61%)	139 (89%)	*NR* (55%)	*NR* (60%)	*NR* (87%)	**65 (91%)**
CIFAR10*	75.0%	*NR*	*NR*	57	*NR*	*NR*	62	**27**
(Label Noise)	85.0%	*NR* (28%)	*NR* (23%)	*NR* (84%)	*NR* (48%)	*NR* (62%)	*NR* (85%)	**49 (91%)**
CIFAR100*	40.0%	138	139	71	*NR*	93	65	**48**
	52.5%	*NR* (42%)	*NR* (42%)	132 (55%)	*NR* (18%)	*NR* (43%)	133 (54%)	**77 (61%)**
CIFAR100*	40.0%	*NR*	*NR*	94	*NR*	89	79	**49**
(Label Noise)	47.5%	*NR* (4%)	*NR* (4%)	142 (48%)	*NR* (14%)	*NR* (43%)	116 (50%)	**65 (60%)**
CINIC10	70.0%	*NR*	*NR*	34	*NR*	*NR*	38	**27**
	77.5%	*NR* (36%)	*NR* (50%)	64 (82%)	*NR* (39%)	*NR* (60%)	97 (80%)	**38 (83%)**
CINIC10	60.0%	*NR*	*NR*	22	*NR*	30	24	**13**
(Label Noise)	67.5%	*NR* (16%)	*NR* (16%)	35 (79%)	*NR* (39%)	*NR* (64%)	38 (78%)	**17 (82%)**
SST2*	82.5%	8	2	3	*NR*	7	**1**	**1**
	90.0%	*NR* (87%)	4 (91%)	*NR* (89.7%)	*NR* (66%)	*NR* (83%)	6 (90%)	**3 (92%)**
CoLA*	75.0%	8	6	16	*NR*	*NR*	34	**3**
	80.0%	*NR* (78%)	*NR* (79%)	*NR* (78%)	*NR* (62%)	*NR* (69%)	*NR* (76%)	**39 (80%)**

severely drop in accuracy when the noise follows the class confusion matrix [Rolnick et al., 2017] and when we add ambiguous images [Mukhoti et al., 2023] (Appendix G.3).

Together, this suggests that noisy points have high loss (and gradient norm), but also high IL and thus low reducible loss. Their IL is high even when the IL model is small as noisy labels cannot be predicted well using the holdout set (*labeled* evaluation set).

Relevant Points. We study how often less relevant points are selected by creating the CIFAR100 Relevance dataset, in which 80% of the data comes from 20% of the classes. This mimics natural distributions of NLP and vision data where most data comes from few object classes, topics, or words [Baayen and Lieber, 1996; Tian et al., 2021]. Concretely, we retain all examples from 20 randomly chosen "high relevance" classes but only 6% of the examples from other, "low relevance" classes. Intuitively, since the high relevance classes have higher $p_{true}(\boldsymbol{x})$ and are 17x more likely to appear at test time, improving their accuracy improves the test accuracy much more than improving the accuracy of less relevant classes.

The loss and gradient norm methods select more points than uniform selection from the low relevance classes (Figure 8.3). In contrast, RHO-LOSS selects somewhat fewer low relevance points, suggesting these classes have high IL. Since the less relevant classes are less abundant in the holdout set (*labeled* evaluation set), both the small

and large IL models have higher loss on them.

Redundant Points. To investigate whether methods select redundant points, we track the percentage of selected points that are already classified correctly. This is only a proxy for redundancy; points that are classified correctly but with low confidence are not fully redundant, since their loss can be further reduced. We control for the different accuracy reached by each method by averaging only over epochs in which test accuracy is lower than the final accuracy reached by the weakest performing method. Figure 8.3 shows that all methods select fewer redundant points than uniform sampling.

8.3.4 Speedup

Finally, we evaluate how much different selection methods speed up training. Recall that the main application area for this chapter is large web-scraped datasets, known for high levels of noise and redundancy. Clothing-1M is such a dataset (§8.3.0). We also include smaller, clean benchmarks from vision (CIFAR-10, CIFAR-100, CINIC-10) and NLP (CoLA, SST-2). Finally, we study if selection functions are robust to the controlled addition of label noise.

Speedup on Clean Data. RHO-LOSS reaches target accuracies in fewer epochs than uniform selection on all datasets (Table 8.2). It also outperforms state-of-the-art methods by a clear margin in terms of speed and final accuracy. On the challenging CoLA language understanding dataset, the speedup over uniform selection exceeds 10x. In Table G.1 (Appendix G.1), we find similar speedups when using no holdout data.

Speedup on Noisy Data. When adding 10% label noise, batch selection with RHO-LOSS achieves greater speedups while, as hypothesized, prior art degrades (Table 8.2). Notably, on noisier data, the speedup over uniform selection grows.

Speedup on Large Web-Scraped Data. On Clothing-1M, loss-based and gradient norm-based selection fail to match uniform selection, suggesting they are not robust to noise. In contrast, RHO-LOSS reaches the highest accuracy that uniform selection achieves during 50 epochs in just 2 epochs and improves final accuracy (72% vs 70%). Notably, this was possible even though the IL model we used has low accuracy (62.2%) and was trained on ca. 10x fewer data. RHO-LOSS also used 2.7x fewer FLOPs to reach the peak accuracy of uniform selection, including the cost of training the IL model (which could be amortized) and despite our implementation being designed to save time, not compute. While Table 8.2 shows results for a Resnet-50, Figure 8.1 includes several additional architectures, with an average speedup of 18x.

8.4 Related Work

Time-Efficient Data Selection. Forward passes for selection can be accelerated using low-precision hardware or parallelization. While backward passes typically require high precision, forward passes can tolerate lower precision [Jouppi et al., 2017; Jiang et al., 2019], especially as we only need the loss (not the activations which would be needed for backpropagation). A forward pass by default requires roughly 3x less time than a forward-backward pass, but this speedup can be increased to a factor around 10x when using the low-precision cores available in modern GPUs and TPUs [Jouppi et al., 2017; Jiang et al., 2019]. Further, prior work uses a set of workers that perform

forward passes on B_t or on the entire dataset asynchronously while the master process trains on recently selected data [Alain et al., 2015].

Compute-Efficient Data Selection. While we limit our scope to comparing selection functions, and we compute them naively, this choice is inefficient in practice. Selection can be made cheaper by reusing losses computed in previous epochs [Loshchilov and Hutter, 2015; Jiang et al., 2019] or training a small model to predict them [Katharopoulos and Fleuret, 2017; Zhang et al., 2019a; Coleman et al., 2020]. Alternatively, core set methods perform selection once before training [Mirzasoleiman et al., 2020; Borsos et al., 2020], although typically with more expensive selection functions.

Data Selection Functions. RHO-LOSS is best understood as an alternative to existing selection functions, which can be categorized by the properties of points they select and whether they use information about labels. "Hard" points are selected both by high loss [Loshchilov and Hutter, 2015; Kawaguchi and Lu, 2020; Jiang et al., 2019] and high prediction uncertainty [Settles, 2010; Li and Sethi, 2006; Coleman et al., 2020]. However, prediction uncertainty does not require labels and can thus be used for active learning. Despite this, they both suffer from the same problem: high loss and high uncertainty can be caused by noisy (in particular, ambiguous) labels. This also applies to selection of points whose labels are easily forgotten during training [Toneva et al., 2019]. Noisy points are avoided by our negative IL baseline and similar methods [Pleiss et al., 2020; Chen et al., 2019]. Points that reduce (expected) holdout loss are also selected for other applications [Killamsetty et al., 2021b; Ren et al., 2018], although using much more computation.

Variance Reduction Methods. Online batch selection is also used to reduce the variance of the gradient estimator computed by SGD [Katharopoulos and Fleuret, 2018, 2017; Johnson and Guestrin, 2018; Alain et al., 2015]. Such methods typically use importance sampling—points with high (approximate) gradient norm are sampled with high probability but then down-weighted in the gradient calculation to de-bias the gradient estimate. Without de-biasing, methods like RHO-LOSS also create selection bias. However, bias can improve test performance, both in theory and practice [Farquhar et al., 2021; Kawaguchi and Lu, 2020].

8.5 Discussion

To reduce excessive training times, we introduce a theoretically grounded selection function that enables substantial speedups on clean data and even larger speedups on noisy and web-scraped data. By illuminating three properties of optimal selection, we hope to motivate new directions in batch selection. However, our selection function should be combined with methods in §8.4 for cheap and fast selection with maximal speedups.

If you understand, things are just as they are. If you do not understand, things are just as they are.

9

Unifying Approaches in Active Learning and Active Sampling

The topic of this chapter is the explicit connection between Bayesian active learning and active sampling, and the connection between these and other recent approaches in data subset selection: amongst them, BADGE [Ash et al., 2020], BAIT [Ash et al., 2021], PRISM[1][Kothawade et al., 2022], SIMILAR[1] [Kothawade et al., 2021], and GraNd [Paul et al., 2021]. Specifically, we connect the acquisition functions used to select informative samples in these approaches to information-theoretic quantities (short: *information quantities*) that are known from Bayesian optimal experiment design [Lindley, 1956; MacKay, 1992b].

By examining how Fisher information and second-order posterior approximations (Gaussian approximations) can be used for estimating information quantities, we develop a unifying perspective and relate these recent methods to information quantities used in Bayesian active learning (and introduced in previous chapters): for active learning, the expected information gain/(Batch)BALD, which we examined in §1.2.4.2 and §4, the (joint) expected predictive information gain from §7; and, for active sampling, the information gain, mentioned in §1.2.6, and the (joint) predictive information gain from §8.

These connections point us towards possible failure modes of above methods and potential extensions in principled ways. Reciprocaly, they also point towards new extensions of what we have introduced in the previous chapters—very exciting!

We examine well-known approximations that lead to last-layer approaches, find a potential estimation bias when using similarity matrices (kernels) in active learning, compare trace and log determinant approximations in regard to batch acquisition pathologies, and trade off weight- and prediction-space methods in principle.

Limitations

It is important to note the limitations of this chapter. We string together different research areas and disparate literature, while focusing on providing an information-theoretic perspective. For a perspective that is focused on kernel methods and Gaussian Process approximations of neural networks, Holzmüller et al. [2022] extensively covers somve of the mentioned works in active learning above in great detail, which provides a useful addition to this chapter.

Hierarchy of Approximations. Although our results employ a hierarchy of approximations, we do not examine the error terms in detail. This is in line with how these approximations are used in deep learning, where the approximations often only provide motivation for useful mechanisms. However, we try to identify where these

approximations might break, enumerate their limitations, and raise several (empirical) research questions for the future.

Log Loss. While many active learning and active sampling methods are motivated independently of the underlying loss, we will remain focused on log losses, such as the common cross-entropy loss or squared error loss (Gaussian error), as these log losses can be viewed through an information-theoretic and probabilistic lens.

Chapter Structure

In §9.2, we look at second-order posterior approximations (Gaussian approximations), which we use to revisit Fisher information, its properties, special cases, and approximations in §9.3. Our contribution here is to summarize results and provide a consistent notation that simplifies reasoning about information quantities, observed information, and Fisher information.

In §9.4, we approximate the information quantities mentioned above using observed information and Fisher information. We provide a comprehensive overview to understand the differences and similarities and make it easier to spot applications of these approximations in the literature. We pay special attention to the limitations: for example, we will see that some approximations that use the trace of the Fisher information do not take redundancies between samples into account. They exhibit the same pathologies as other methods that, in essence, score points individually (§5). In §9.5, we expand our approach to approximations that use similarity matrices of log-loss gradients. Our contributions are a comprehensive overview of the approximations and the connection to similarity matrices.

In §9.6, we show that (Batch-)BALD and EPIG on the one hand; and BADGE, BAIT, PRISM[1], and SIMILAR[1] on the other hand can be seen as optimizing the same objectives. The difference is that (Batch-)BALD (§4; Houlsby et al. [2011]) and EPIG (§7) operate in prediction space, while Fisher information-based methods operate in weight space: we show that an approximation of EPIG, a transductive active learning objective, using Fisher information, matches the BAIT objective [Ash et al., 2021]. Similarly, we show how BADGE [Ash et al., 2020] approximates the EIG, using the connection to similarity matrices. Finally, we find that *submodularity*-based approaches [Iyer et al., 2021] such as SIMILAR [Kothawade et al., 2021] and PRISM [Kothawade et al., 2022], which report their best results using the log determinant of similarity matrices, approximate information quantities when they perform best. We also show that gradient-length-based methods like EGL [Settles et al., 2007] and GraNd [Paul et al., 2021] can be connected to information quantities.

9.1 Setting

This section some additional notation, concepts, and probabilistic model that we use in this chapter.

Transductive Acquisition Functions. When an acquisition function in data subset selection uses (additional) data $\mathcal{D}^{\text{eval}}$, unlabeled or labeled, to guide acquisitions, we refer to the objective as a *transductive* objective [Yu et al., 2006; Wang et al., 2021].

[1] using log determinant objectives

Active Learning. To increase label efficiency, instead of labeling data indiscriminately, active learning iteratively selects and acquires labels for the *most informative* unlabeled data from a *pool set* $\mathcal{D}^{\text{pool}}$ according to some *acquisition function*. An acquisition function scores the informativeness of an unlabeled candidate sample $\boldsymbol{x}^{\text{acq}}$, and the sample that maximizes this score is selected for labeling. After each acquisition step, the model is retrained to take the newly labeled data into account. Labels can be acquired individually or in batches (*batch acquisition*, see below). The *expected information gain (EIG)*

$$\mathrm{I}[\Omega; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}] \tag{EIG/BALD}$$

and *(joint) expected predictive information gain (JEPIG and EPIG, respectively)*

$$\mathrm{I}[\{Y_i^{\text{eval}}\}; Y^{\text{acq}} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \boldsymbol{x}^{\text{acq}}], \tag{JEPIG}$$
$$\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid X^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] \tag{EPIG}$$

are examples of such acquisition functions. EPIG and JEPIG are transductive acquisition functions as they depend on $X^{\text{eval}}, \{\boldsymbol{x}_i^{\text{eval}}\}$, respectively.

Active Sampling. To increase training efficiency, instead of training with all samples, active sampling (sometimes also called *data pruning*) [Paul et al., 2021] selects the most informative sample $(\boldsymbol{x}^{\text{acq}}, y^{\text{acq}})$ from the training set to train on next. This can be done statically before training the model, in which case this is also referred to as core-set selection, or dynamically, in which case it is also referred to as curriculum learning. The *information gain (IG)*

$$\mathrm{I}[\Omega; y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}], \tag{IG}$$

and *(joint) predictive information gain (JPIG or PIG, respectively)*

$$\mathrm{I}[\{y_i^{\text{eval}}\}; y^{\text{acq}} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \boldsymbol{x}^{\text{acq}}], \tag{JPIG}$$
$$\mathrm{I}[Y^{\text{eval}}; y^{\text{acq}} \mid X^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] \quad (= \mathbb{E}_{\hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}^{\text{eval}}, y^{\text{eval}})} \mathrm{I}[y^{\text{eval}}; y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}]) \tag{PIG}$$

are examples of such acquisition functions. PIG and JPIG are transductive acquisition functions.

Submodular Acquisition Functions. Choosing the subset $\{\boldsymbol{x}_i^{\text{acq}}\}$ naively is intractable due to the exponential number of possible acquisition batches. Instead of maximizing the acquisition function on all possible batches, we can often use submodularity [Nemhauser et al., 1978]. A set function f is submodular when:

$$f(A \cup B) \leq f(A) + f(B) - f(A \cap B). \tag{submodular}$$

An acquisition batch $\{\boldsymbol{x}_i^{\text{acq}}\}$ can be constructed greedily by selecting the samples that increase the acquisition function the most one-by-one. This greedy algorithm is guaranteed to find a $1 - \frac{1}{e}$-optimal acquisition batch for monotone submodular acquisition functions.

Although the EIG is (monotone) submodular, leading to efficient batch acquisition (§4), the other information quantities (IG, EPIG, JEPIG, PIG) are usually not submodular. We examine the details of this and compare to the relevant literature in §9.6.

Table 9.1: *Taxonomy of Information Quantities for Data Subset Selection.* In general, information quantities can be split into ones for active sampling or active learning, into non-transductive and transductive ones, and in the transductive case, into taking an expectation or the joint over (additional) evaluation samples. Here, we show the information quantities for individual acquisition. For batch acquisition, $\{Y_i^{\text{acq}}\}, \{y_i^{\text{acq}}\}, \{\boldsymbol{x}_i^{\text{acq}}\}$ can be substituted.

		Active Learning		Active Sampling	
Non-Transductive		EIG/BALD	$\mathrm{I}[\Omega; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}]$	IG	$\mathrm{I}[\Omega; y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}]$
Transductive (using $\mathcal{D}^{\text{eval}}$)	Expectation	EPIG	$\mathbb{E}_{\hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}^{\text{eval}})} \mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}]$	PIG	$\mathbb{E}_{\hat{\mathrm{p}}_{\text{true}}(y^{\text{eval}}, \boldsymbol{x}^{\text{eval}})} \mathrm{I}[y^{\text{eval}}; y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}]$
	Joint	JEPIG	$\mathrm{I}[\{y_i^{\text{eval}}\}; Y^{\text{acq}} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \boldsymbol{x}^{\text{acq}}]$	JPIG	$\mathrm{I}[\{y_i^{\text{eval}}\}; y^{\text{acq}} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \boldsymbol{x}^{\text{acq}}]$

Taxonomy of Information Quantities. Table 9.1 shows the information quantities along three dimensions: active learning vs active sampling, non-transductive vs transductive, and taking the expectation vs the joint over evaluation samples for transductive information quantities.

9.2 Second-Order Posterior Approximation

Laplace approximations are a standard tool in Bayesian statistics and machine learning [Daxberger et al., 2021; Immer et al., 2021]. In this section, we review the Laplace approximation and introduce it as a special case of a more flexible second-order posterior approximation, a *Gaussian approximation*. It is central to approximating information quantities using observed information, defined in this section, and Fisher information, defined in §9.3.

Our goal is to approximate the posterior $\mathrm{p}(\omega \mid \mathcal{D}, \mathcal{D}^{\text{train}})$ using a (multivariate) Gaussian distribution, where $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ are additional (new) samples, and we start with $\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$ as the "prior" distribution—we will drop $\mathcal{D}^{\text{train}}$ and use $\mathrm{p}(\omega)$ when possible, to shorten the notation.

To begin, we complete the square of a second-order Taylor approximation around the log-parameter likelihood for a fixed ω^*:

$$
\begin{aligned}
&\log \mathrm{p}(\omega) \\
&\quad \approx \log \mathrm{p}(\omega^*) + \nabla_\omega[\log \mathrm{p}(\omega^*)](\omega - \omega^*) + \frac{1}{2}(\omega - \omega^*)^T \nabla_\omega^2[\log \mathrm{p}(\omega^*)](\omega - \omega^*) \qquad (9.1) \\
&\quad = \frac{1}{2}(\omega - (\omega^* - \nabla_\omega^2[\log \mathrm{p}(\omega^*)]^{-1} \nabla_\omega[\log \mathrm{p}(\omega^*)])^T \\
&\qquad \nabla_\omega^2[\log \mathrm{p}(\omega^*)] \\
&\qquad (\omega - (\omega^* - \nabla_\omega^2[\log \mathrm{p}(\omega^*)]^{-1} \nabla_\omega[\log \mathrm{p}(\omega^*)])) \\
&\qquad + \dots . \qquad (9.2)
\end{aligned}
$$

Importantly, we can express this more concisely by extending the notation of $\mathrm{H}[\cdot]$ to its derivatives:

Notation 9.1. We write $\mathrm{H}'[\cdot]$ for the Jacobian and $\mathrm{H}''[\cdot]$ for the Hessian of $\mathrm{H}[\cdot]$:

$$\mathrm{H}'[\cdot] \triangleq -\nabla_\omega \log \mathrm{p}(\cdot), \qquad (9.3)$$

$$\mathrm{H}''[\cdot] \triangleq -\nabla_\omega^2 \log \mathrm{p}(\cdot). \qquad (9.4)$$

This notation will be helpful throughout this chapter, as both observed information and Fisher information can be expressed in terms of the Hessian of the negative log-parameter likelihood. The Jacobian of the entropy is also known as score function.

Then, we can write:

$$\mathrm{H}[\omega] \approx \mathrm{H}[\omega^*] + \mathrm{H}'[\omega^*](\omega - \omega^*) + \frac{1}{2}(\omega - \omega^*)^T \, \mathrm{H}''[\omega^*] \, (\omega - \omega^*) \tag{9.5}$$

$$= \frac{1}{2}(\omega - (\omega^* - \mathrm{H}''[\omega^*]^{-1} \, \mathrm{H}'[\omega^*])^T \, \mathrm{H}''[\omega^*] \, (\omega - (\omega^* - \mathrm{H}''[\omega^*]^{-1} \, \mathrm{H}'[\omega^*])) + \dots . \tag{9.6}$$

Comparing this to the information content of a multivariate Gaussian distribution:

$$\mathrm{H}[\mathcal{N}(w;\, \mu,\, \Sigma)] = \frac{1}{2}(\omega - \mu)^T \, \Sigma^{-1} \, (\omega - \mu) + \dots , \tag{9.7}$$

we obtain the Gaussian approximation, which we will apply throughout this chapter:

Proposition 9.2. *The* Gaussian approximation *of the distribution* $\mathrm{p}(\omega)$ *of* Ω *around some* ω^* *is given by:*

$$\Omega \approx \mathcal{N}(\omega^* - \mathrm{H}''[\omega^*]^{-1} \, \mathrm{H}'[\omega^*],\, \mathrm{H}''[\omega^*]^{-1}), \tag{9.8}$$

where $\mathrm{H}''[\omega^*]$ *must be positive-definite. If* ω^* *is also a (global) minimizer of* $\mathrm{H}[\omega]$ *(that is,* $\mathrm{H}'[\omega^*] = 0$*), we obtain the* Laplace approximation*:*

$$\Omega \approx \mathcal{N}(\omega^*,\, \mathrm{H}''[\omega^*]^{-1}). \tag{9.9}$$

Approximation Quality. However, this approximation can be arbitrarily bad depending on $\mathrm{p}(\omega)$ and ω^*. Given enough data, it is often argued that $\mathrm{p}(\omega)$ will concentrate around the maximum a posteriori (MAP) estimate, giving rise to the Laplace approximation. In statistics, the Bernstein-von Mises theorem is often used to motivate this, but insufficient data to reach concentration of parameters and multimodality in over-parameterized models [Long, 2022] can be an issue for deep active learning and active sampling.

Flat Minimum Intuition. A positive definite Hessian implies that the information content (point-wise entropy) is convex around ω^* and, equivalently, that the (log) posterior is concave around ω^*. The latter provides an intuition for the Gaussian approximation: the Hessian measures curvature, and the "flatter" the Hessian, e.g., the smaller the largest eigenvalue or the smaller the determinant, the less the loss changes when ω^* is perturbed. This leads to the search for flat minima as a way to improve generalization [Hinton and van Camp, 1993; Hochreiter and Schmidhuber, 1994; Smith and Le, 2018].

Notation 9.3. To further shorten the notation, we write $\mathrm{H}''[\mathcal{D} \,|\, \omega^*]$ instead of $\mathrm{H}''[\{y_i\} \,|\, \{\boldsymbol{x}_i\}, \omega^*]$.

Posterior Approximation of $\Omega \,|\, \mathcal{D}$. While the Laplace approximation is centered on a (global) minimizer, the Gaussian approximation can be used for a (potentially

low-quality) posterior approximation in general. We can expand $\mathrm{H}[\omega^* \mid \mathcal{D}]$ using Bayes' theorem and the additivity of the logarithm. That is, we have:

$$\mathrm{H}[\omega^* \mid \mathcal{D}] = \mathrm{H}[\mathcal{D} \mid \omega^*] + \mathrm{H}[\omega^*] - \mathrm{H}[\mathcal{D}], \tag{9.10}$$

and then, as $\mathrm{H}[\mathcal{D}]$ is independent of ω:

$$\mathrm{H}'[\omega^* \mid \mathcal{D}] = \mathrm{H}'[\mathcal{D} \mid \omega^*] + \mathrm{H}'[\omega^*] + 0 = \mathrm{H}'[\mathcal{D} \mid \omega^*] + \mathrm{H}'[\omega^*], \tag{9.11}$$

$$\mathrm{H}''[\omega^* \mid \mathcal{D}] = \mathrm{H}''[\mathcal{D} \mid \omega^*] + \mathrm{H}''[\omega^*]. \tag{9.12}$$

Proposition 9.4. *The* observed information $\mathrm{H}''[\{y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*]$ *is additive:*

$$\mathrm{H}''[\{y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*] = \sum_i \mathrm{H}''[y_i \mid \boldsymbol{x}_i, \omega^*] = \sum_i -\nabla^2_\omega \log \mathrm{p}(y_i \mid \boldsymbol{x}_i, \omega^*). \tag{9.13}$$

Note that the observed information has the opposite sign compared to other works because it simplifies the exposition.

Uninformative Prior. For a Gaussian prior $\mathrm{p}(\omega) \sim \mathcal{N}(\mu, \Sigma)$, we have $\mathrm{H}''[\omega^*] = \Sigma^{-1}$ and $\mathrm{H}''[\omega^* \mid \mathcal{D}] = \mathrm{H}''[\mathcal{D} \mid \omega^*] + \Sigma^{-1}$. For an uninformative prior with "infinite prior variance" $\Sigma^{-1} \to 0$, we have $\mathrm{H}''[\omega^*] = 0$, and $\mathrm{H}''[\omega^* \mid \mathcal{D}] = \mathrm{H}''[\mathcal{D} \mid \omega^*]$.

Proposition 9.5. *The entropy of the second-order approximation of* $\mathrm{p}(\omega)$ *around* ω^* *is*

$$\mathrm{H}[\Omega] \approx -\tfrac{1}{2} \log \det \mathrm{H}''[\omega^*] + C_k, \tag{9.14}$$

where $C_k = \frac{k}{2} \log 2\pi e$ *is a constant (independent of* $\mathcal{D}$ *and* ω^**) and* k *is the number of dimensions of* ω*.*

While Proposition 9.5 is straightforward, it is the main result for this section as it will allow us to approximate all the mentioned information quantities in §9.4 and §9.5.

9.3 Fisher Information

Fisher information plays a central role in the approximations of information quantities because, unlike the observed information, it is always positive semi-definite. We use Fisher information to unify various acquisition functions in §9.6. The following section revisits Fisher information, its properties, special cases, and common approximations. All proofs are given in §H.1.

In particular, we look at two special cases with more favorable properties: following Kunstner et al. [2019], when we can write our model as $\mathrm{p}(y \mid \hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}; \omega))$, where $\hat{f}(\boldsymbol{x}; \omega)$ are the logits, and $\mathrm{p}(y \mid \hat{\boldsymbol{z}})$ is a distribution from the exponential family, Fisher information is independent of y, which has useful consequences as we shall see; and following Chaudhuri et al. [2015], when we have a *Generalized Linear Model (GLM)*, observed information also is independent of y. The results for the GLM are often applied as an approximation known as *Generalized Gauss-Newton approximation (GGN)*. Together with numerical approximations, such as a diagonal approximation or low-rank factorizations, observed information and Fisher information can then be efficiently approximated for large deep neural networks [Daxberger et al., 2021].

Definition 9.1. The *Fisher information* $\mathrm{H}''[Y \mid \boldsymbol{x}, \omega^*]$ is the expectation over observed information using the model's own predictions $\mathrm{p}(y \mid \boldsymbol{x}, \omega^*)$ for a given $\boldsymbol{x}$ at ω^*:

$$\mathrm{H}''[Y \mid \boldsymbol{x}, \omega^*] = \mathbb{E}_{\mathrm{p}(y|\boldsymbol{x},\omega^*)}[\mathrm{H}''[y \mid \boldsymbol{x}, \omega^*]]. \tag{9.15}$$

This notation of the Fisher information is consistent with the notation for information quantities we have used far (§1.2.1 and §2), but extended to the observed information: the Fisher is but an expectation over the observed information, and the observed information is the Hessian of the negative log-likelihood.

Proposition 9.6. *Like observed information, Fisher information is additive:*

$$\mathrm{H}''[\{Y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*] = \sum_i \mathrm{H}''[\{Y_i\} \mid x_i, \omega^*]. \tag{9.16}$$

There are two other equivalent definitions of Fisher information:

Proposition 9.7. *Fisher information is equivalent to:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\mathrm{H}'[y \mid x, \omega^*]^T \, \mathrm{H}'[y \mid x, \omega^*]] = \mathrm{Cov}[\mathrm{H}'[Y \mid x, \omega^*]]. \tag{9.17}$$

Special Case: Exponential Family. Kunstner et al. [2019] show in their appendix that if we split a discriminative model into prelogits $\hat{f}(\boldsymbol{x}; \omega)$ and a predictor $\mathrm{p}(y \mid \hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}; \omega))$, Fisher information does not depend on y when $\mathrm{p}(y \mid \hat{\boldsymbol{z}})$ is a distribution from an exponential family (independent of ω). This covers a normal distribution for regression parameterized by mean and variance predictions or a categorical distribution via the softmax function. The following statements and proofs follow Kunstner et al. [2019]:

Proposition 9.8. *The Fisher information* $\mathrm{H}''[Y \mid \boldsymbol{x}, \omega^*]$ *for a model* $\mathrm{p}(y \mid \hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}; \omega^*))$ *is equivalent to:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \nabla_\omega \hat{f}(x; \omega^*)^T \, \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\nabla^2_{\hat{\boldsymbol{z}}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x; \omega^*)]] \nabla_\omega \hat{f}(x; \omega^*), \tag{9.18}$$

where $\nabla^2_{\hat{\boldsymbol{z}}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}; \omega^*)]$ *is short for* $\nabla^2_{\hat{\boldsymbol{z}}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}}]|_{\hat{\boldsymbol{z}}=\hat{f}(\boldsymbol{x};\omega^*)}$.

Proposition 9.9. *The Fisher information* $\mathrm{H}''[Y \mid \boldsymbol{x}, \omega^*]$ *of a model of the form* $\mathrm{p}(y \mid \hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}; \omega^*))$ *is independent of* y*, where* $\mathrm{p}(y \mid \hat{\boldsymbol{z}})$ *is a distribution from an exponential family, i.e.,* $\log \mathrm{p}(y \mid \hat{\boldsymbol{z}}) = \hat{\boldsymbol{z}}^T T(y) - A(\hat{\boldsymbol{z}}) + \log h(y)$*:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \nabla_\omega \hat{f}(x; \omega^*)^T \, \nabla^2_{\hat{\boldsymbol{z}}} A(\hat{\boldsymbol{z}} = \hat{f}(x; \omega^*)) \, \nabla_\omega \hat{f}(x; \omega^*). \tag{9.19}$$

It is crucial that the exponential distribution not depend on ω. This simplifies computing Fisher information: no expectation over ys is needed anymore. The full outer product may not be needed explicitly either.

As examples, we will consider two common parameteric distributions from the exponential family:

Gaussian Distribution. When $\mathrm{p}(y \mid \hat{\boldsymbol{z}}) = \mathcal{N}(y; \hat{\boldsymbol{z}}, 1)$, we have $\mathrm{H}''[y \mid \hat{\boldsymbol{z}}] = 1$ for all $y, \hat{\boldsymbol{z}}$, and thus

$$\mathrm{H}''[Y \mid x, \omega^*] = \nabla_\omega \hat{f}(x; \omega^*)^T \, \nabla_\omega \hat{f}(x; \omega^*). \tag{9.20}$$

Categorical Distribution. When $\mathrm{p}(y \mid \hat{\boldsymbol{z}}) = \mathrm{softmax}(\hat{\boldsymbol{z}})_y$, we have $\mathrm{H}''[y \mid \hat{\boldsymbol{z}}] = \mathrm{diag}(\pi) - \pi\,\pi^T$, with $\pi_y = \mathrm{p}(y \mid \hat{\boldsymbol{z}})$, and thus:

$$\mathrm{H}''[Y \mid x, \omega^*] = \nabla_\omega \hat{f}(x;\omega^*)^T \left(\mathrm{diag}(\pi) - \pi\,\pi^T\right) \nabla_\omega \hat{f}(x;\omega^*). \tag{9.21}$$

Special Case: Generalized Linear Models. Chaudhuri et al. [2015] require that observed information is independent of y, which we will also use later. This holds for Generalized Linear Models:

Definition 9.2. A *generalized linear model (GLM)* is a model $\mathrm{p}(y \mid \hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x};\omega))$ such that $\log \mathrm{p}(y \mid \hat{\boldsymbol{z}}) = \hat{\boldsymbol{z}}^T T(y) - A(\hat{\boldsymbol{z}}) + \log h(y)$ is a distribution of the exponential family, independent of ω, and $\hat{f}(\boldsymbol{x};\omega) = \omega^T \boldsymbol{x}$ is linear in the parameters ω.

Proposition 9.10. *The observed information* $\mathrm{H}''[y \mid \boldsymbol{x}, \omega^*]$ *of a GLM is independent of* y.

$$\mathrm{H}''[y \mid x, \omega^*] = \nabla_\omega \hat{f}(x;\omega^*)^T \nabla^2_{\hat{\boldsymbol{z}}} \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x;\omega^*)] \nabla_\omega \hat{f}(x;\omega^*) \tag{9.22}$$

$$= \nabla_\omega \hat{f}(x;\omega^*)^T \nabla^2_{\hat{\boldsymbol{z}}} A(w^T x) \nabla_\omega \hat{f}(x;\omega^*). \tag{9.23}$$

Proposition 9.11. *For a model such that the observed information* $\mathrm{H}''[y \mid \boldsymbol{x}, \omega^*]$ *is independent of* y*, we have:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \mathrm{H}''[y^* \mid x, \omega^*] \tag{9.24}$$

for any y^**, and also trivially:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \mathbb{E}_{\mathrm{p}(y|x)}[\mathrm{H}''[y \mid x, \omega^*]]. \tag{9.25}$$

Note that the expectation is over $\mathrm{p}(y|\boldsymbol{x})$ and not $\mathrm{p}(y|\boldsymbol{x}, \omega^*)$, and $\mathbb{E}_{\mathrm{p}(\{y_i\}|\{\boldsymbol{x}_i\})}[\mathrm{H}''[\{y_i\}| \{\boldsymbol{x}_i\}, \omega^*]] = \mathrm{H}''[\{Y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*]$ is additive then.

Proposition 9.12. *For a GLM, when* $\hat{f}(\boldsymbol{x};\omega) : \mathbb{R}^D \to \mathbb{R}^{\mathtt{C}}$*, where* $\mathtt{C}$ *is the number of classes (outputs),* D *is the number of input dimensions,* $\omega \in \mathbb{R}^{D \times \mathtt{C}}$*, and assuming the parameters are flattened into a single vector for the Jacobian, we have* $\nabla_\omega \hat{f}(\boldsymbol{x};\omega) = \mathrm{Id}_{\mathtt{C}} \otimes \boldsymbol{x}^T \in \mathbb{R}^{\mathtt{C} \times (\mathtt{C} \cdot D)}$*, where* $\otimes$ *denotes the Kronecker product, and:*

$$\nabla_\omega \hat{f}(x;\omega^*)^T \nabla^2_{\hat{\boldsymbol{z}}} A(\omega^T x) \nabla_\omega \hat{f}(x;\omega^*) = \nabla^2_{\hat{\boldsymbol{z}}} A(w^T x) \otimes x\,x^T. \tag{9.26}$$

This property is useful for computing the Fisher information of a GLM in practice [Ash et al., 2021].

$\mathrm{p}(\mathbf{y} \mid \mathbf{x}, \omega^*)$ vs $\mathrm{p}(\mathbf{y} \mid \mathbf{x})$. Having a GLM solves an important issue we will encounter in §9.4: approximating the EIG requires taking an expectation over $\mathrm{p}(y \mid \boldsymbol{x})$ and not $\mathrm{p}(y \mid \boldsymbol{x}, \omega^*)$. One can approximate $\mathrm{p}(y \mid \boldsymbol{x}) \approx \mathrm{p}(y \mid \boldsymbol{x}, \omega^*)$, which can be justified in the limit, but this is probably not a good approximation in the cases interesting for active learning and active sampling. With a GLM, this is not a problem.

Generalized Gauss-Newton Approximation. In the case of an exponential family but not a GLM, the equality in Proposition 9.10 is often used as an approximation for the observed information—we simply use the respective Fisher information as an approximation of observed information (via Proposition 9.9):

$$\mathrm{H}''[y \mid x, \omega^*] \approx \mathrm{H}''[Y \mid x, \omega^*] = \nabla_\omega \hat{f}(x; \omega^*)^T \nabla^2_{\hat{z}} A(w^T x) \nabla_\omega \hat{f}(x; \omega^*). \tag{9.27}$$

This is known as *Generalized Gauss-Newton (GGN) approximation* [Kunstner et al., 2019; Immer et al., 2021]. This approximation has the advantage that it is always positive semi-definite unlike the true Hessian.

Last-Layer Approaches. GLMs are often used in deep active learning [Ash et al., 2020, 2021; Kothawade et al., 2022, 2021]. If we split the model into $\mathrm{p}(y \mid \boldsymbol{x}, \omega) = \mathrm{p}(y \mid \boldsymbol{z} = \omega^T f(\boldsymbol{x}))$, where $\boldsymbol{z} = f(\boldsymbol{x})$ are the embeddings and treat the encoder $f(\boldsymbol{x})$ as fixed, we obtain a GLM based on the weights of the last layer, which uses the embeddings as input.

Armed with this knowledge, we can now derive approximations for the information quantities of interest using observed information and Fisher information and consider their properties. The GGN approximation and last-layer approaches feature heavily in the literature to make computing these approximations more tractable as they reduce computational requirements and memory usage.

9.4 Approximating Information Quantities

We now derive approximations and proxy objectives for information quantities. We base them on observed information and Fisher information introduced in the previous sections. These approximations help us connect the information quantities to existing the literature in non-Bayesian data subset selection in §9.6.

In particular, we derive approximations for EIG and EPIG as they show the qualitative differences between non-transductive and transductive objectives, and compare the approximations of the IG and EIG: importantly, there is no difference between the latter when we use a GLM or the GGN approximation. This covers two of the three dimensions in Table 9.1. We examine JEPIG and the other quantities in the appendix in §H.2. We find that the trace approximations of the EPIG and JEPIG objective matches, suggesting that using the trace approximations might be too loose an approximation to capture important qualities of EPIG (§7). Additional derivations and details can also be found in §H.2. All this leads to Figure 9.1, which relates the different approximations to each other and shows that they follow simple patterns.

9.4.1 Approximate Expected Information Gain

The expected information gain is a popular acquisition function in Bayesian optimal experimental design [Lindley, 1956] and in active learning, where it is also known as BALD [Houlsby et al., 2011; Gal et al., 2017].

We can approximate the EIG $\mathrm{I}[\Omega; \{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}]$ of acquisition candidates $\{\boldsymbol{x}_i^{\mathrm{acq}}\}$ using Gaussian approximations:

$$\mathrm{I}[\Omega; \{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}]$$

$$= \mathrm{H}[\Omega] - \mathrm{H}[\Omega \mid \{Y_i^{\mathrm{acq}}\}, \{\boldsymbol{x}_i^{\mathrm{acq}}\}] \tag{9.28}$$
$$= \mathrm{H}[\Omega] - \mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[\mathrm{H}[\Omega \mid \{y_i^{\mathrm{acq}}\}, \{\boldsymbol{x}_i^{\mathrm{acq}}\}]] \tag{9.29}$$
$$\approx -\tfrac{1}{2}\log\det \mathrm{H}''[\omega^*] - \mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[-\tfrac{1}{2}\log\det \mathrm{H}''[\omega \mid \{y_i^{\mathrm{acq}}\}, \{\boldsymbol{x}_i^{\mathrm{acq}}\}]] \tag{9.30}$$
$$= \tfrac{1}{2}\,\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[\log\det\left((\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*] + \mathrm{H}''[\omega^*])\,\mathrm{H}''[\omega^*]^{-1}\right)] \tag{9.31}$$
$$= \tfrac{1}{2}\,\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[\log\det\left(\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]\ \mathrm{H}''[\omega^*]^{-1} + Id\right)]. \tag{9.32}$$

using Proposition 9.5 twice, where the constant C_k cancels out in Equation 9.30 as we subtract two entropy terms.

Generalized Linear Model. When we have a GLM, we can use Proposition 9.11 to obtain:

$$\mathrm{I}[\Omega; \{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}] \tag{9.33}$$
$$\approx \ldots = \tfrac{1}{2}\,\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[\log\det\left(\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]\ \mathrm{H}''[\omega^*]^{-1} + Id\right)] \tag{9.34}$$
$$= \tfrac{1}{2}\log\det\left(\mathrm{H}''[\{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]\ \mathrm{H}''[\omega^*]^{-1} + Id\right). \tag{9.35}$$

We can upper-bound the log determinant and obtain:

$$\leq \tfrac{1}{2}\,\mathrm{tr}\left(\mathrm{H}''[\{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]\ \mathrm{H}''[\omega^*]^{-1}\right) \tag{9.36}$$
$$= \tfrac{1}{2}\sum_i \mathrm{tr}\left(\mathrm{H}''[\{Y_i^{\mathrm{acq}}\} \mid \boldsymbol{x}^{\mathrm{acq}}{}_i, \omega^*]\ \mathrm{H}''[\omega^*]^{-1}\right). \tag{9.37}$$

where we have used the following inequality (proof in §H.2.1):

Lemma 9.13. *For symmetric, positive semi-definite matrices A, we have (with equality iff $A = 0$):*

$$\log\det(A + Id) \leq \mathrm{tr}(A). \tag{9.38}$$

General Case & Exponential Family. For the general case, we need to make a strong approximation:

$$\mathrm{p}(\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}) \approx \mathrm{p}(\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*), \tag{9.39}$$

which might hold for a mostly converged posterior but probably not in cases with little data. This turns the approximation into an upper bound. Alternatively, we could use the GGN approximation when we have an exponential family for the same result (but not an upper bound). See §H.2.1 for the derivation.

Proposition 9.14 (EIG). *The expected information gain can be approximately upper bounded via:*

$$\begin{aligned}
&\mathrm{I}[\Omega; \{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \\
&\quad \overset{\approx}{\leq} \tfrac{1}{2} \log\det\left(\sum_i \mathrm{H}''[{Y^{acq}}_i \mid {\boldsymbol{x}^{acq}}_i, \omega^*]\ \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]^{-1} + Id\right) && (9.40) \\
&\quad \leq \tfrac{1}{2} \sum_i \mathrm{tr}\left(\mathrm{H}''[{Y^{acq}}_i \mid {\boldsymbol{x}^{acq}}_i, \omega^*]\, \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]^{-1}\right). && (9.41)
\end{aligned}$$

Furthermore, we have the following proxy objective:

$$\arg\max_{\{\boldsymbol{x}_i^{acq}\}} \mathrm{I}[\Omega; \{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] = \arg\max_{\{\boldsymbol{x}_i^{acq}\}} -\mathrm{H}[\Omega \mid \{Y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}], \quad (9.42)$$

with

$$\begin{aligned}
&-\mathrm{H}[\Omega \mid \{Y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \\
&\quad \overset{\approx}{\leq} \tfrac{1}{2} \log\det\left(\sum_i \mathrm{H}''[{Y^{acq}}_i \mid {\boldsymbol{x}^{acq}}_i, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]\right) - C_k. && (9.43)
\end{aligned}$$

The second statement follows from Equation 9.28, since $\mathrm{H}[\Omega \mid \mathcal{D}^{\text{train}}]$ is constant and provides a proxy objective when we are only interested in optimizing the EIG. In §9.6, we connect it to the expected gradient length approach in active learning and show that an ablation in Ash et al. [2021] examines the wrong objective.

Batch Acquisition Pathologies. Importantly, this approximation of the EIG using the trace is additive, whereas the one using the log determinant is not. This means that the trace approximation ignores the dependencies between the samples and can only lead to naive top-k batch acquisition; see **??** and §5 for details of the pathologies of top-k batch acquisition.

9.4.2 Approximate Information Gain

Following the same steps, we can also approximate the information gain, which is useful for active sampling:

Proposition 9.15 (IG). *The* information gain $\mathrm{I}[\Omega; \{y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] = \mathrm{H}[\Omega \mid \mathcal{D}^{\text{train}}] - \mathrm{H}[\Omega \mid \{y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}]$ *can be approximately upper bounded via:*

$$\begin{aligned}
&\mathrm{I}[\Omega; \{y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \\
&\quad \approx \tfrac{1}{2} \log\det\left(\mathrm{H}''[\{y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] \; \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]^{-1} + Id\right) && (9.44) \\
&\quad \leq \tfrac{1}{2} \sum_i \mathrm{tr}\left(\mathrm{H}''[y^{acq}{}_i \mid \boldsymbol{x}^{acq}{}_i, \omega^*] \; \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]^{-1}\right). && (9.45)
\end{aligned}$$

Furthermore, we have the following proxy objective:

$$\arg\max_{\{\boldsymbol{x}_i^{acq}\}} \mathrm{I}[\Omega; \{y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] = \arg\max_{\{\boldsymbol{x}_i^{acq}\}} -\mathrm{H}[\Omega \mid \{y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \tag{9.46}$$

with

$$-\mathrm{H}[\Omega \mid \{y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \approx \tfrac{1}{2} \log\det\left(\mathrm{H}''[\{y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]\right) - C_k. \tag{9.47}$$

Comparison to EIG. Importantly, when we have a GLM or use the GGN approximation, this approximation of the IG is equal to the one of the EIG. This tells us that active learning on a GLM with the EIG approximation will work as well as if we had access to the labels. Equivalently, active sampling via IG with the GGN approximation will not work better than the respective active learning approach.

9.4.3 Approximate (Joint) Expected Predictive Information Gain

In transductive active learning, we have access to an (empirical) distribution $\hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}^{\text{eval}})$, e.g., the pool set, and want to find the $\{\boldsymbol{x}_i^{\text{acq}}\}$ that maximize the *expected predictive information gain* from §7). The approximations here will help us connect BAIT [Ash et al., 2021] to EPIG. For simplicity, we consider the non-batch case here. The batch case can be handled analogously. The EPIG objective is defined as:

$$\arg\max_{\boldsymbol{x}^{\text{acq}}} \mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid X^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] = \arg\max_{\boldsymbol{x}^{\text{acq}}} \mathbb{E}_{\hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}^{\text{eval}})} \mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}], \tag{9.48}$$

We expand the objective as follows:

$$\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid X^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] = \mathrm{I}[\Omega; Y^{\text{eval}} \mid X^{\text{eval}}] - \mathrm{I}[\Omega; Y^{\text{eval}} \mid X^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}], \tag{9.49}$$

where $\mathrm{I}[\Omega; Y^{\text{eval}} \mid X^{\text{eval}}]$ can be removed from the objective because it is independent of $\boldsymbol{x}^{\text{acq}}$. Thus, optimizing EPIG is equivalent to *minimizing* $\mathrm{I}[\Omega; Y^{\text{eval}} \mid X^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}]$:

$$\arg\max_{\boldsymbol{x}^{\text{acq}}} \mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid X^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] = \arg\min_{\boldsymbol{x}^{\text{acq}}} \mathrm{I}[\Omega; Y^{\text{eval}} \mid X^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}]. \tag{9.50}$$

Following Proposition 9.14, this can be approximated by:

$$\begin{aligned}
&\mathrm{I}[\Omega; Y^{\text{eval}} \mid X^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}] \\
&\quad \approx \tfrac{1}{2} \mathbb{E}_{\mathrm{p}(y^{\text{eval}}, y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}) \, \hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}^{\text{eval}})} && (9.51) \\
&\qquad \left[\log\det\left(\mathrm{H}''[y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, \omega^*] \left(\mathrm{H}''[y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*] + \mathrm{H}''[\omega^*]\right)^{-1} + Id\right)\right]. && (9.52)
\end{aligned}$$

Generalized Linear Model. For a generalized linear model, we can drop the expectation and obtain:

$$\begin{aligned}
&\mathrm{I}[\Omega; Y^{\mathrm{eval}} \mid X^{\mathrm{eval}}, Y^{\mathrm{acq}}, \boldsymbol{x}^{\mathrm{acq}}] \\
&\quad\approx \tfrac{1}{2}\,\mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{\mathrm{eval}})}[\log\det\left(\mathrm{H}''[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \omega^*]\,(\mathrm{H}''[Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^*])^{-1} + Id\right)] && (9.53)\\
&\quad\leq \tfrac{1}{2}\log\det\left(\mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{\mathrm{eval}})}[\mathrm{H}''[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \omega^*]]\,(\mathrm{H}''[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^*])^{-1} + Id\right) && (9.54)\\
&\quad\leq \tfrac{1}{2}\mathrm{tr}\left(\mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{\mathrm{eval}})}\left[\mathrm{H}''[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \omega^*]\right]\,(\mathrm{H}''[Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^*])^{-1}\right), && (9.55)
\end{aligned}$$

where we have used the concavity of the log determinant and Lemma 9.13.

General Case & Exponential Family. To our knowledge, there is no rigorous way to obtain a similar result in the general case as the Fisher information for an acquisition candidate now lies within an inverted term. Of course, the GGN approximation can be applied when we have an exponential family, which leads to the GLM result above as an approximation. See §H.2.2 for more details.

Proposition 9.16 (EPIG)**.** *For a generalized linear model (or with the GGN approximation), we have:*

$$\arg\max_{\{\boldsymbol{x}_i^{acq}\}} \mathrm{I}[Y^{eval}; \{Y_i^{acq}\} \mid X^{eval}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\mathrm{train}}] = \arg\min_{\{\boldsymbol{x}_i^{acq}\}} \mathrm{I}[\Omega; Y^{eval} \mid X^{eval}, \{Y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\mathrm{train}}], \tag{9.56}$$

with

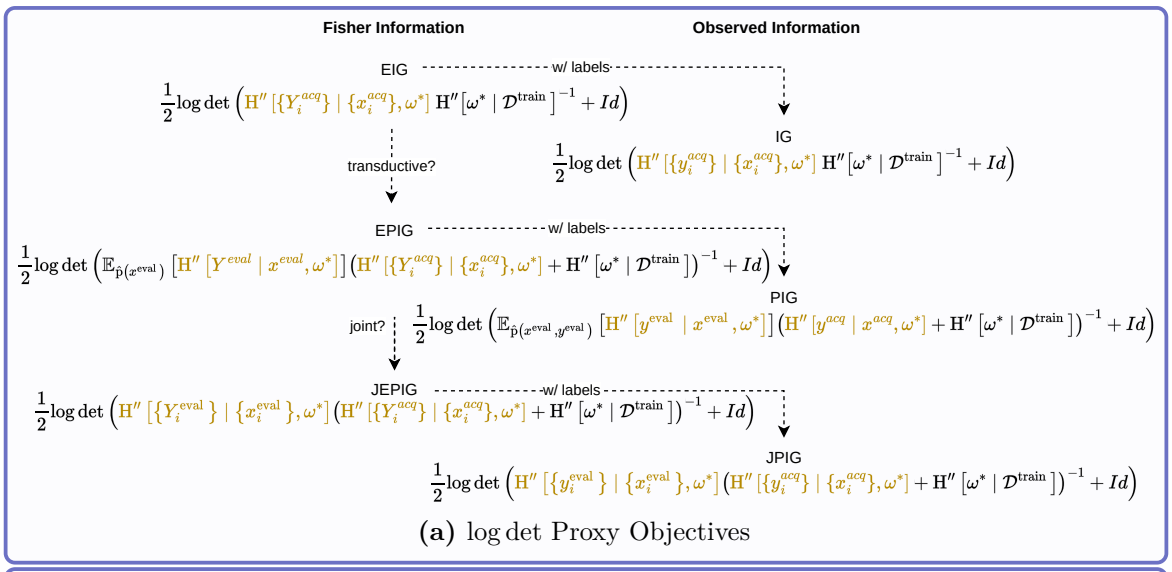
$$\begin{aligned}
&\mathrm{I}[\Omega; Y^{eval} \mid X^{eval}, \{Y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\mathrm{train}}] \\
&\quad\approx \mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{eval})}[\log\det\left(\mathrm{H}''[Y^{eval} \mid \boldsymbol{x}^{eval}, \omega^*]\,(\mathrm{H}''[\{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^*])^{-1} + Id\right)] && (9.57)\\
&\quad\leq \tfrac{1}{2}\log\det\left(\mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{eval})}[\mathrm{H}''[Y^{eval} \mid \boldsymbol{x}^{eval}, \omega^*]]\,(\mathrm{H}''[\{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}])^{-1} + Id\right) && (9.58)\\
&\quad\leq \tfrac{1}{2}\mathrm{tr}\left(\mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{eval})}\left[\mathrm{H}''[Y^{eval} \mid \boldsymbol{x}^{eval}, \omega^*]\right]\,(\mathrm{H}''[\{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}])^{-1}\right). && (9.59)
\end{aligned}$$

Batch Acquisition Pathologies. Unlike for the EIG, the trace approximation of EPIG is not additive in $\{\boldsymbol{x}_i^{\mathrm{acq}}\}$, and we cannot conclude that it suffers from batch acquisition pathologies like the trace approximation of the EIG.

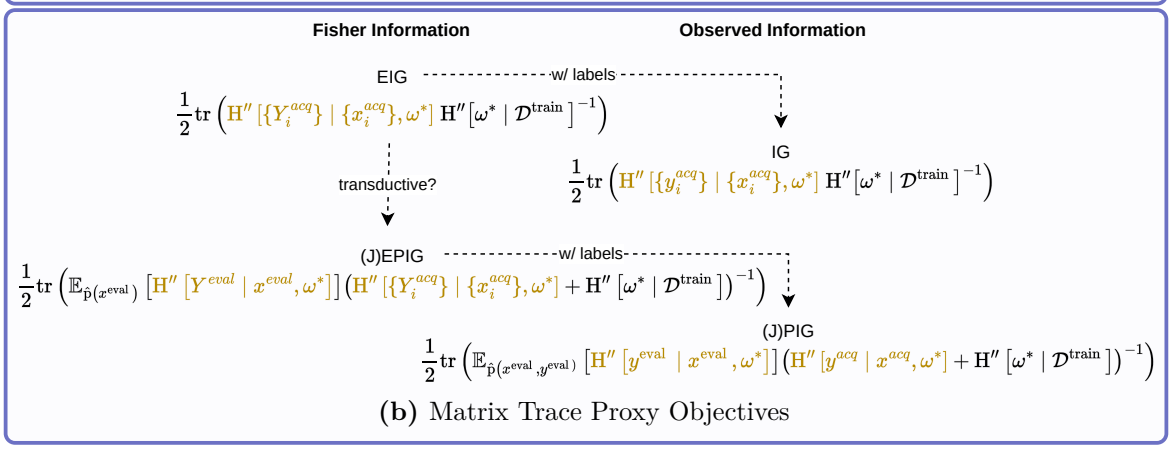
Approximations for JEPIG, PIG and JPIG. We can follow the same derivation for JEPIG:

$$\begin{aligned}
&\mathrm{I}[\Omega; \{Y_i^{\mathrm{eval}}\} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, Y^{\mathrm{acq}}, \boldsymbol{x}^{\mathrm{acq}}] && (9.60)\\
&\quad\approx \tfrac{1}{2}\log\det\left(\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{eval}}\}, y^{\mathrm{acq}} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \boldsymbol{x}^{\mathrm{acq}})}[\mathrm{H}''[\{y_i^{\mathrm{eval}}\} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \omega^*]]\,(\mathrm{H}''[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^*])^{-1} + Id\right).
\end{aligned}$$

Then, applying the steps after Equation 9.54, we can devise similar approximations. PIG and JPIG follow the same pattern. Details can be found in §H.2.3 and §H.2.4. But how do all these approximations relate to each other?



(a) log det Proxy Objectives



(b) Matrix Trace Proxy Objectives

Figure 9.1: *Comparison of the Approximations/Proxy Objectives.* The difference between active learning and active sampling objectives is in using the Fisher information, which is label independent, or the observed information, which uses label information. JEPIG and EPIG have equivalent proxy objectives when using the matrix trace; see §H.2.4. The notation makes it obvious that in the GLM case, or when the GGN approximation is used, active learning and active sampling approximations match as $\mathrm{H}''[y \mid \boldsymbol{x}, \omega^*] = \text{ (resp. } \approx) \, \mathrm{H}''[Y \mid \boldsymbol{x}, \omega^*]$.

9.4.4 Comparison of the Different Information Quantity Approximations

Figure 9.1 compares the different information quantity approximations for both the log-determinant and trace approximations. We empirically compare the approximations with prediction-space methods in §H.5. Importantly, the trace approximations of (E)PIG match those of J(E)PIG up to a constant factor (unlike the log-determinant approximations); see §H.2.4 for details.

In §9, we argued that JEPIG converges to BALD in the data limit of the evaluation set—when there are no outliers in the pool set— while EPIG does not. The trace approximation is too strong to preserve this difference. Does this difference matter in practice? We leave this for future work.

Crucially, for GLMs or when using the GGN approximation, the respective active learning and active sampling objectives (EIG and IG, etc.) are equivalent as Fisher

information and observed information are the same. In contrast, in the general case, the approximations for EPIG and JEPIG do not have a principled derivation.

9.5 Similarity Matrices and One-Sample Approximations

Many data subset selection methods [Iyer et al., 2021; Kothawade et al., 2022, 2021; Ash et al., 2020] use similarity matrices of the loss Jacobians $\mathrm{H}'[\hat{y} \mid \boldsymbol{x}]$ (gradient kernels), where $\hat{y}$ is usually a hypothesized pseudo-label: often the arg max prediction of the model for $\boldsymbol{x}$. Here, we connect such similarity matrices to the Fisher information and the approximations of information quantities from §9.4. The proofs are given in §H.3. Together with §9.4, this section provides a unified framework for understanding the approximations of information quantities using Fisher information and lays the foundation for the next section, which will connect the cited works in §9 to the approximations of information quantities.

Connection to Fisher Information. Crucially, given $\mathcal{D} = \{(y_i, \boldsymbol{x}_i)_i\}$, if we let

$$\hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*] \triangleq \begin{pmatrix} \vdots \\ \mathrm{H}'[y_i \mid x_i, \omega^*] \\ \vdots \end{pmatrix} \tag{9.61}$$

be a "data matrix" of the Jacobians, then $\hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]\hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]^T$ gives the similarity matrix $S[\mathcal{D} \mid \omega^*]$ using the Euclidean inner product:

$$S[\mathcal{D} \mid \omega]_{ij} \triangleq \langle \mathrm{H}'[y_i \mid \boldsymbol{x}_i, \omega^*], \mathrm{H}'[y_j \mid \boldsymbol{x}_j, \omega^*] \rangle = \hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]\hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]^T. \tag{9.62}$$

Sampling $\{y_i\} \sim \mathrm{p}(\{y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*)$, the "flipped" product $\hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]^T \hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]$ yields a *one-sample estimate* of the Fisher information $\mathrm{H}''[\{Y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*]$:

$$\mathrm{H}''[\{Y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*] = \sum_i \mathrm{H}''[Y_i \mid x_i, \omega^*] = \mathbb{E}_{\mathrm{p}(\{y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*)} \sum_i \mathrm{H}'[y_i \mid x_i, \omega^*]^T \mathrm{H}'[y_i \mid x_i, \omega^*] \tag{9.63}$$

$$= \mathbb{E}_{\mathrm{p}(\{y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*)} \hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]^T \hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]. \tag{9.64}$$

Hard Pseudo-Labels. Importantly, using the arg max class for y_i, we only obtain a biased estimate [Kunstner et al., 2019, §B].

Connection to the Expected Information Gain. When we define an inner product $\langle \cdot, \cdot \rangle_{\mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]}$ using the Hessian, we can connect the similarity matrix, which uses this inner product:

$$S_{\mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]}[\mathcal{D} \mid \omega^*] \triangleq \hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]\, \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]^{-1} \hat{\mathrm{H}}'[\mathcal{D} \mid \omega^*]^T, \tag{9.65}$$

to our information gain approximations.

Specifically, we apply the matrix-determinant lemma $\det(AB+M) = \det M \det(Id + BM^{-1}A)$ to obtain:

Proposition 9.17. *Given $\mathcal{D}^{\text{train}}$, $\{\boldsymbol{x}_i^{acq}\}$ and (sampled) $\{y_i^{acq}\}$, we have for the EIG:*

$$\mathrm{I}[\Omega; \{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \overset{\approx}{\leq} \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^*|\mathcal{D}^{\text{train}}]}[\mathcal{D}^{\text{acq}} \mid \omega^*] + Id\right) \tag{9.66}$$

$$\leq \tfrac{1}{2}\operatorname{tr} S_{\mathrm{H}''[\omega^*|\mathcal{D}^{\text{train}}]}[\mathcal{D}^{\text{acq}} \mid \omega^*] \tag{9.67}$$

Proposition 9.18. *Assuming an uninformative posterior $\mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}] = \lambda Id$ for $\lambda \to 0$, and given $\mathcal{D}^{\text{train}}$, $\{\boldsymbol{x}_i^{acq}\}$, and (sampled) $\{y_i^{acq}\}$, we have for the EIG (before taking $\lambda \to 0$):*

$$\mathrm{I}[\Omega; \{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \overset{\approx}{\leq} \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{acq}} \mid \omega^*] + \lambda Id\right) - \tfrac{|\mathcal{D}^{\text{acq}}|}{2}\log\lambda. \tag{9.68}$$

As the second term is independent of $\{\boldsymbol{x}_i^{acq}\}$, we can use the following proxy objective in the limit:

$$\frac{1}{2}\log\det\left(S[\mathcal{D}^{\text{acq}} \mid \omega^*]\right). \tag{9.69}$$

Connection to Other Approximate Information Quantities. Interestingly, we can use the above to obtain approximations of the predictive information gains (EPIG and JEPIG) because the terms that would tend towards $-\infty$ cancel out; see §H.3 for details. For EPIG, we have:

Proposition 9.19. *Given $\mathcal{D}^{\text{eval}}, \mathcal{D}^{\text{train}}$, $\{\boldsymbol{x}_i^{acq}\}$ and (sampled) $\{y_i^{acq}\}$, we have for the EPIG:*

$$\begin{aligned}&\mathrm{I}[Y^{eval}; \{Y_i^{acq}\} \mid X^{eval}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}]\\ &\approx \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^*|\mathcal{D}^{\text{train}}]}[\mathcal{D}^{\text{eval}} \mid \omega^*] + Id\right) - \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^*|\mathcal{D}^{\text{train}}]}[\mathcal{D}^{\text{acq}}, \mathcal{D}^{\text{eval}} \mid \omega^*] + Id\right)\\ &\quad + \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^*|\mathcal{D}^{\text{train}}]}[\mathcal{D}^{\text{acq}} \mid \omega^*] + Id\right).\end{aligned} \tag{9.70}$$

For an uninformative prior, we have:

$$\approx \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{eval}} \mid \omega^*]\right) - \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{acq}}, \mathcal{D}^{\text{eval}} \mid \omega^*]\right) + \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{acq}} \mid \omega^*]\right). \tag{9.71}$$

We can drop the terms that only depend on $\mathcal{D}^{\text{eval}}$ when we are interested in proxy objectives for optimization.

These results help connect the objectives of PRISM and SIMILAR to the EIG and EPIG in the next section.

9.6 Information Quantities in Prior Literature

Now, we can connect approaches in non-Bayesian literature to information quantities. Additional proofs are given in §H.4.

9.6.1 BAIT, ActiveSetSelect, and (J)EPIG

BAIT in "Gone Fishing" [Ash et al., 2021], ActiveSetSelect in "Convergence Rates of Active Learning for Maximum Likelihood Estimation" [Chaudhuri et al., 2015], and (J)EPIG (§7) approximate the same objective.

Ash et al. [2021] introduce the *BAIT* objective for deep active learning:

$$\arg\min_{\{\boldsymbol{x}_i^{\text{acq}}\}} \operatorname{tr}\left(\left(\mathrm{H}''[\{Y_i^{\text{acq}}\} \mid \{\boldsymbol{x}_i^{\text{acq}}\}, \omega^*] + \mathrm{H}''[Y^{\text{train}} \mid \boldsymbol{x}^{\text{train}}, \omega^*] + \lambda I\right)^{-1} \mathrm{H}''[\{Y_i^{\text{eval}}\} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \omega^*]\right), \tag{BAIT}$$

where λ is a hyperparameter[2].

BAIT is based on a similar objective for GLMs from Chaudhuri et al. [2015]. While Ash et al. [2021] apply this objective to DNNs, they only use the last layer to approximate the Fisher information. The last layer, with appropriate activation functions and losses, constitutes a GLM as seen in §9.3.

Following Proposition 9.16, we immediately see that Ash et al. [2021] perform transductive active learning (using the pool set as an evaluation set) and approximate a proxy objective for (J)EPIG:

Proposition 9.20. *Both Ash et al. [2021] and Chaudhuri et al. [2015] perform transductive active learning, approximating (J)EPIG (§7) using a last-layer approach (or GLM):*

$$\begin{aligned}&\arg\max_{\boldsymbol{x}^{acq}} \mathrm{I}[Y^{eval}; \{Y_i^{acq}\} \mid X^{eval}, \{\boldsymbol{x}_i^{acq}\}] \\ &\quad\approx \arg\min_{\boldsymbol{x}^{acq}} \operatorname{tr}(\mathrm{H}''[\{Y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \omega^*] \, (\mathrm{H}''[\{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}])^{-1}),\end{aligned} \tag{9.72}$$

with $\mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}] = \mathrm{H}''[\mathcal{D}^{\text{train}} \mid \omega^*] + \mathrm{H}''[\omega^*]$ *and* $\mathrm{H}''[\omega^*] = \lambda \, Id$.

Proof. This follows immediately for GLM (last-layer approaches) when we expand $\mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]$. Chaudhuri et al. [2015] in particular uses an uninformative prior, that is $\lambda = 0$. Comparing the resulting objectives yields the statement. □

Thus, Ash et al. [2021] and we in §7 employ the same underlying acquisition function, albeit using very different approaches: Ash et al. [2021] use a last-layer Fisher information matrix, whereas we use approximate BNNs and sample joint predictions in §7.

Research Questions. EPIG is not submodular, and the greedy selection of an acquisition batch does not come with any optimality guarantees. While we sidesteps this in §7 by focusing on individual acquisitions mainly, Ash et al. [2021] propose a heuristic that empirically performs better: They greedily select additional acquisition candidates in forward pass (twice the intended batch acquisition size) and then greedily remove the least informative samples from the batch in a backward pass. *Would this heuristic also prove beneficial for all the other information quantities that are not submodular?*

While Ash et al. [2021] state that they only use the last-layer approach for performance reasons, following §9.3, it does not seem that this approach translates beyond a last-layer approach for DNNs in a principled fashion (see §H.2.2). *Is there a principled approach for the general case that goes beyond last-layer active learning when using Fisher information without the GGN approximation?*

[2]This is the BAIT objective as computed in Algorithm 1 in Ash et al. [2021] and in the published implementation https://github.com/JordanAsh/badge/blob/master/query_strategies/bait_sampling.py.

Ash et al. [2021] ablate trace and determinantal approaches, similar to comparing Equation 9.59 and Equation 9.58, yet they do not include $+Id$ in the log determinant expression, which leads them to examine the EIG in their ablation[3]:

$$\begin{aligned}
&\arg\min_{\boldsymbol{x}^{\text{acq}}} \log\det(\mathrm{H}''[\{Y_i^{\text{eval}}\} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \omega^*] \,(\sum_i \mathrm{H}''[{Y^{\text{acq}}}_i \mid {\boldsymbol{x}^{\text{acq}}}_i, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}])^{-1}) \\
&= \arg\min_{\boldsymbol{x}^{\text{acq}}} \log\det \mathrm{H}''[\{Y_i^{\text{eval}}\} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \omega^*] - \log\det(\sum_i \mathrm{H}''[{Y^{\text{acq}}}_i \mid {\boldsymbol{x}^{\text{acq}}}_i, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]) \qquad (9.73) \\
&= \arg\max_{\boldsymbol{x}^{\text{acq}}} \log\det(\sum_i \mathrm{H}''[{Y^{\text{acq}}}_i \mid {\boldsymbol{x}^{\text{acq}}}_i, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]) + C, \qquad (9.74)
\end{aligned}$$

and the last term matches the EIG in Equation 9.58 up constant terms independent of $\boldsymbol{x}^{\text{acq}}$. Thus, the ablations in Ash et al. [2021] only compares EIG and EPIG. *Could comparing Equation 9.59 and Equation 9.58 provide more insightful results about the trade-offs between trace and determinant approximations?*

9.6.2 BADGE and BatchBALD

BADGE [Ash et al., 2020] performs batch acquisition using a similarity matrix: Using the concepts of §9.5, BADGE uses hard pseudo-labels together with last-layer gradient embeddings for the similarity matrix $S[\mathcal{D}^{\text{acq}} \mid \omega^*]$. The authors sample from a k-DPP [Kulesza and Taskar, 2011] based on this similarity matrix to select a diverse batch of samples for acquisition. However, to further speed up acquisitions, BADGE uses k-MEANS++ [Arthur and Vassilvitskii, 2007; Ostrovsky et al., 2012] instead of a k-DPP: it uses the Jacobians $\mathrm{H}'[y \mid \boldsymbol{x}, \omega^*]$ of the data matrix directly and samples a diverse batch based on the Euclidean distance between these Jacobians. However, sampling from k-DPPs does not pick the most informative batch overall, and the ablations in Ash et al. [2020] show that k-MEANS++ outperforms k-DPP. Finally, the paper only motivates using gradient embeddings with hard pseudo-labels through intuitions: the gradient length captures information about the model's uncertainty, and diverse update directions capture information about the model's diversity [Ash et al., 2020]. The paper makes no explicit connection to information theory.

Following Proposition 9.17, since BADGE can be seen as using a last-layer approach for the similarity matrix $S[\mathcal{D}^{\text{acq}} \mid \omega^*]$ with hard pseudo-labels, BADGE approximates $\mathrm{I}[\Omega; \{Y_i^{\text{acq}}\} \mid \{\boldsymbol{x}_i^{\text{acq}}\}, \mathcal{D}^{\text{train}}]$ with an uninformative posterior distribution:

Proposition 9.21. *BADGE maximizes an approximation of the EIG with an uninformative prior.*

Comparison to BatchBALD. Similarly, BatchBALD (§4) approximates the EIG in the batch acquisition case but by using prediction-space samples. Moreover, BatchBALD uses a greedy approach to select batch candidates instead of sampling via a k-DPP or k-MEANS++.

As the EIG is submodular, determining the acquisition batch is a submodular optimization problem and, therefore, can be solved by greedy selection with $1 - \frac{1}{e}$ optimality [Nemhauser et al., 1978].

[3] Ash et al. [2021] accidentally writes arg max instead of arg min in §5.1 in their paper, but c.f. Algorithm 1 with the trace objective. Algorithm 2 & 3 in §B in the appendix use the correct final objective.

Research Questions. Hard pseudo-labels lead to biased estimates. *Would one-sample estimates perform better? And could greedy batch selection work better than sampling via a k-DPP?* This would be closer to the batch acquisition strategy followed by BatchBALD.

9.6.3 SIMILAR and PRISM

Based on Iyer et al. [2021], Kothawade et al. [2021] and Kothawade et al. [2022] investigate *submodular active learning* for DNNs: they take an *information function* f, which is a non-negative, montone/non-decreasing, submodular function (and then is also subadditive as a consequence):

$$\begin{aligned} f(A) &\geq 0, && \text{(non-negative)} \\ f(A) &\leq f(B) \text{ for } A \subseteq B, && \text{(monotone)} \\ f(A \cup B) &\leq f(A) + f(B) - f(A \cap B), && \text{(submodular)} \\ f(A \cup B) &\leq f(A) + f(B) && \text{(subadditive)} \end{aligned}$$

for all $A, B \subseteq \mathcal{D}^{\text{pool}}$ and define a "*submodular conditional gain*" and an "*submodular (conditional) mutual information*" as

$$H_f(A \mid B) \triangleq f(A \cup B) - f(B) \tag{9.75}$$

$$I_f(A; B) \triangleq f(A \cup B) - f(A) - f(B). \tag{9.76}$$

For $f(\{\boldsymbol{x}_i\}) = \mathrm{H}[\{Y_i\} \mid \{\boldsymbol{x}_i\}]$, this simply yields the regular information quantities. Hence, Kothawade et al. [2021] and Kothawade et al. [2022] examine other information functions and submodular quantities in the context of active learning: amongst them set covers, graph cuts, facility location, and log determinants (LogDet) of similarity matrices. Like BADGE [Ash et al., 2020], the similarity matrix uses hard pseudo-labels. Like BatchBALD (§4), they use a greedy approach for acquisition [Nemhauser et al., 1978].

Using our results, we immediately see that the LogDet objective, which we can write as $\log\det S[\mathcal{D}^{\text{acq}} \mid \omega^*]$, exactly matches the EIG approximation in §9.18; furthermore, in §H.4.1, we show that the LogDetMI objective matches an approximation of JEPIG (and similarly, derive the LogDetCMI objective as well):

Proposition 9.22. *The LogDet objective* $\log\det S[\mathcal{D}^{\text{acq}} \mid \omega^*]$ *is an approximation of the EIG and the LogDetMI objective*

$$\log\det S[\mathcal{D}^{\text{acq}} \mid \omega^*] - \log\det(S[\mathcal{D}^{\text{acq}} \mid \omega^*] - S[\mathcal{D}^{\text{acq}}; \mathcal{D}^{\text{eval}} \mid \omega^*] S[\mathcal{D}^{\text{eval}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{eval}}; \mathcal{D}^{\text{acq}} \mid \omega^*]) \tag{9.77}$$

is an approximation of a proxy objective for EPIG, where we use $S[\mathcal{D}_1; \mathcal{D}_2 \mid \omega^*]$ *to denote the (non-symmetric) similarity matrix between* $\mathcal{D}_1$ *and* $\mathcal{D}_2$.

Notably, the experimental results for the LogDet-based quantities are reported as among the best in Kothawade et al. [2021] and Kothawade et al. [2022]. As such, since the LogDet quantities approximate Shannon's information quantities (which are not explicitly examined in those works), the promising experimental results compared to other submodular information functions support the hypothesis that approximating Shannon's information quantities works well in active learning and active sampling.

Research Questions. Similar to BADGE, the scores are biased by using hard pseudo-labels. *Could one-sample estimates perform better?* Furthermore, as LogDetMI is not submodular, *could the approach from BAIT of expanding and shrinking the acquisition batch in a forward and backward pass improve performance here as well?*

9.6.4 Expected Gradient Length

The *Expected Gradient Length (EGL)* [Settles et al., 2007; Settles, 2010] is an acquisition function in active learning and is usually defined for non-Bayesian models. Originally, it was an expectation over the gradient norm. In more recent literature [Huang et al., 2016], it is introduced using the squared gradient norm:

$$\mathbb{E}_{\mathrm{p}(y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*)} \left\| \mathrm{H}'[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] \right\|^2 . \tag{EGL}$$

Using a diagonal approximation of Fisher information, we show in §H.4.2:

Proposition 9.23. *The EIG for a candidate sample $\boldsymbol{x}^{acq}$ approximately lower-bounds the EGL:*

$$2\,\mathrm{I}[\Omega; Y^{acq} \mid \boldsymbol{x}^{acq}] \overset{\approx}{\leq} \mathbb{E}_{\mathrm{p}(y^{acq} \mid \boldsymbol{x}^{acq}, \omega^*)} \left\| \mathrm{H}'[y^{acq} \mid \boldsymbol{x}^{acq}, \omega^*] \right\|^2 + \mathit{const.} \tag{9.78}$$

9.6.5 Deep Learning on a Data Diet

In active sampling, Paul et al. [2021] use the gradient length of given labeled samples $\boldsymbol{x}, y$ (averaged over multiple training runs) as an acquisition function to select the most informative samples from the training set to speed up training:

$$\mathbb{E}_{\mathrm{q}(\omega)} \left\| \mathrm{H}'[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega] \right\|^2 , \tag{GraNd}$$

which they call the *gradient norm score (GraNd)*. The expectation is taken over the model parameters at initialization or after training for a few epochs—as this is not easily expressed using a posterior distribution, we use $\mathrm{q}(w)$ to denote the distribution.

Proposition 9.24. *The IG for a candidate sample $\boldsymbol{x}^{acq}$ approximately lower-bounds the gradient norm score (GraNd) at ω^* up to a second-order term:*

$$2\,\mathrm{I}[\Omega; y^{acq} \mid \boldsymbol{x}^{acq}] \overset{\approx}{\leq} \mathbb{E}_{\mathrm{q}(\omega)}[\left\| \mathrm{H}'[y^{acq} \mid \boldsymbol{x}^{acq}, \omega] \right\|^2] - \mathbb{E}_{\mathrm{q}(\omega)}[\mathrm{tr}\left(\frac{\nabla_\omega^2\, \mathrm{p}(y \mid x, \omega)}{\mathrm{p}(y \mid x, \omega)} \right)] + \mathit{const.} \tag{9.79}$$

The second term might not be negligible. Hence, GraNd (the first term on the left) might deviate from the information gain. *How does the information gain compares to GraNd in practice?*

We provide a reproducibility analysis in Appendix B.1, which helped discover a now fixed bug in Paul et al. [2021].

9.7 Discussion

We have examined Fisher information and Gaussian approximations and have derived weight-space approximations for various information quantities. This has allowed us to connect these information quantities to objectives already used in the literature. Moreover, we can make the following concluding points:

Last-Layer Approaches. Methods that only use last-layer Fisher information or similar perform data subset selection on embeddings only, despite feature learning being arguably the most important strength of deep neural networks. However, these approaches can find great use with large pre-trained models, which are only fine-tuned on new data domains [Tran et al., 2022].

Bias in Hard Pseudo-Labels. Several methods (BADGE, PRISM, SIMILAR) use hard pseudo-labels for computing the similarity matrices, leading to biased estimates. Can one-sample estimates perform better? Can the equivalent approximations that do not make use of similarity matrices perform better?

Trace vs $\log\det$ **Approximations.** We have presented a hierarchy of approximations and bounds. Ablating these approximations along multiple dimensions, including whether to use the trace or $\log\det$ approximation and whether to use a GLM, the GGN approximation or the full Fisher information, could provide interesting insights into what is attainable.

Batch Acquisition Pathologies. Approaches that use the matrix trace instead of the log determinant can end up being additive for batch candidates $\{\boldsymbol{x}_i^{\text{acq}}\}$ and, therefore, by definition, cannot take redundancies between batch candidates into account, leading to failures detailed in §4 and §5. This is an issue for trace approximations of the (E)IG. Does the trace approximation of (JE)PIG handle batch acquisition pathologies better? Similarly, most information quantities are not submodular, yet we use a greedy algorithm to select the acquisition batches. Is the heuristic proposed by Ash et al. [2021] generally beneficial?

Weight vs. Prediction Space. BADGE, BAIT, and the LogDet objectives of PRISM and SIMILAR [Ash et al., 2020, 2021; Kothawade et al., 2021, 2022] approximate information quantities in weight space, while (Batch)BALD, (J)EPIG, and (J)PIG (§4, §7 and §8) approximate the information quantities in prediction space. Both approaches have their limitations:

Weight-space approaches can suffer from the Gaussian approximation being of low quality: the Laplace approximation only captures the posterior distribution well once it concentrates sufficiently. However, this is unlikely to happen in a low-data regime.

Prediction-space approaches can suffer from a combinatorial explosion as the batch acquisition size increases because prediction configurations have to be enumerated or sampled to approximate the information quantities. In addition, many parameter samples might be needed to obtain low-variance estimates.

Importantly, prediction space approaches also require drawing samples from the posterior distribution but do not estimate the information quantities using the posterior distribution, unlike weight space approaches.

Informativeness Scores. Taking a step back, we have seen that a Bayesian perspective using information quantities connects seemingly disparate literature. Although Bayesian methods are often seen as separate from (non-Bayesian) active learning and active sampling, the sometimes fuzzy notion of "informativeness" expressed through various different objectives in non-Bayesian settings collapses to the same couple of information quantities, which were, in principle, already known by Lindley [1956] and MacKay [1992b].

Old paths tread anew,
In light of insights accrued,
Understanding grew.

10

Black-Box Batch Active Learning for Regression

By selectively acquiring labels for a subset of available unlabeled data, active learning [Atlas et al., 1989] is suited for situations where the acquisition of labels is costly or time-consuming, such as in medical imaging or natural language processing. However, in deep learning, many recent batch active learning methods have focused on *white-box approaches* that rely on the model being parametric and differentiable and which use first or second-order derivatives (e.g. model embeddings)[1].

This can present a limitation in real-world scenarios where model internals or gradients might not be accessible—or might be expensive to access. This is particularly true in the case of 'foundation models' [Bommasani et al., 2021] and large language models such as GPT-3 [Brown et al., 2020], for example, when accessed via a third party. More generally, a lack of differentiability might hinder application of white-box batch active learning approaches to non-differentiable models.

To address these limitations, we examine *black-box batch active learning (B^3AL)* for regression which is compatible with a wider range of machine learning models. By *black-box*, we mean that our approach only relies on model predictions and does not require access to model internals or gradients. Our approach is rooted in Bayesian principles and only requires model predictions from a small (bootstrapped) ensemble. Specifically, we utilize an *(empirical) predictive covariance kernel* based on sampled predictions. We show that the well-known gradient kernel [Kothawade et al., 2021, 2022; Ash et al., 2020, 2021] can be seen as an approximation of this predictive covariance kernel.

The proposed approach extends to non-differentiable models through a Bayesian view on the hypothesis space formulation of active learning, based on the ideas behind query-by-committee [Seung et al., 1992]. This enables us to use batch active learning methods, such as BAIT [Ash et al., 2021] and BADGE [Ash et al., 2020] in a black-box setting with non-differentiable models, such as random forests or gradient-boosted trees.

We evaluate black-box batch active learning on a diverse set of regression datasets. Unlike the above *white-box* parametric active learning methods which scale in the number of (last-layer) model parameters or the embedding size, our method scales in the number of drawn predictions, and we show that we can already obtain excellent results with a small ensemble. Our results demonstrate the label efficiency of B^3AL for various machine learning models. For deep learning models, B^3AL even performs better than the corresponding state-of-the-art white-box methods in many cases.

[1] Model embeddings can also be seen as first-order derivatives of the model score under regression in regard to the last layer.

We focus on regression as classification using the same approach would require Laplace approximations, Monte Carlo sampling, or expectation propagation [Williams and Rasmussen, 2006; Hernández-Lobato et al., 2011]. This complicates a fair comparison as translating existing classification methods in active learning is not straightforward. For the same reasons, we also do not consider proxy-based active learning methods [Coleman et al., 2020], which constitutes an orthogonal direction to our investigation.

In summary, by leveraging the strengths of kernel-based methods and Bayesian principles, our approach improves the labeling efficiency of a range of differentiable and non-differentiable machine-learning models with surprisingly strong performance.

The rest of the chapter is organized as follows: in §10.1, we discuss related work in active learning and kernel-based methods. In §10.2, we describe the relevant background and provide a detailed description of B³AL. In §10.3, we detail the experimental setup and provide the results of our experimental evaluation. Finally, §10.4 concludes with a discussion and directions for future research.

10.1 Related Work

Differentiable Models . Many acquisition functions approximate well-known information-theoretic quantities [MacKay, 1992b], often by approximating Fisher information implicitly or explicitly. This can be computationally expensive, particularly in deep learning where the number of model parameters can be large—even when using last-layer approximations or assuming a generalized linear model (§9). BADGE [Ash et al., 2020] and BAIT [Ash et al., 2021] approximate the Fisher information using last-layer loss gradients or the Hessian, respectively, but still have a computational cost scaling with the number of last layer weights. This also applies to methods using similarity matrices (kernels) based on loss gradients of last-layer weights such as SIMILAR [Kothawade et al., 2022] and PRISM [Kothawade et al., 2022], to only name a few. Importantly, all of these approaches require differentiable models.

Non-Differentiable Models. *Query-by-committee (QbC, Seung et al. [1992])* measures the disagreement between different model instances to identify informative samples and has been applied to regression [Krogh and Vedelsby, 1994; Burbidge et al., 2007]. Kee et al. [2018] extend QbC to the batch setting with a diversity term based on the distance of data points in input space. Nguyen et al. [2012] show batch active learning for random forests. They train an ensemble of random forests and evaluate the joint entropy of the predictions of the ensemble for batch active learning, which can be seen as a special case of BatchBALD in regression.

BALD. This chapter most closely aligns with the BALD-family of Bayesian active learning acquisition functions [Houlsby et al., 2011], which focus on classification tasks, however. The crucial insight of BALD is applying the symmetry of mutual information to compute the expected information gain in prediction space instead of in parameter space. As a result, BALD is a *black-box technique* that only leverages model predictions. The estimators utilized by Gal et al. [2017] and also the ones for BatchBALD from §4 enumerate over all classes, leading to a trade-off between combinatorial explosion and Monte-Carlo sampling, which can result in degraded quality estimates as acquisition batch sizes increase. Houlsby et al. [2011], Gal et al. [2017] have not applied BALD to regression tasks.

Kernel-Based Methods. Holzmüller et al. [2022] examine the previously mentioned methods and unify them using gradient-based kernels. Specifically, they express BALD [Houlsby et al., 2011], BatchBALD (§4), BAIT [Ash et al., 2021], BADGE [Ash et al., 2020], ACS-FW [Pinsler et al., 2019], and Core-Set [Sener and Savarese, 2018]/FF-Active [Geifman and El-Yaniv, 2017] using kernel-based methods for regression tasks. They also propose a new method, LCMD (largest cluster maximum distance).

Comparison to This Chapter. This chapter extends the work of Houlsby et al. [2011] and Holzmüller et al. [2022] by combining the prediction-based approach with a kernel-based formulation. This trivially enables batch active learning on regression tasks using black-box predictions for a wide range of existing batch active learning methods.

10.2 Methodology

In this chapter, we focus on regression, which is a common task in machine learning. We assume that the target y is real-valued ($\in \mathbb{R}$) with homoscedastic Gaussian noise:

$$Y \mid \boldsymbol{x}, \boldsymbol{\omega} \sim \mathcal{N}(\mu(\boldsymbol{x}; \boldsymbol{\omega}), \sigma_N^2). \tag{10.1}$$

Equivalently, $Y \mid \boldsymbol{x}, \boldsymbol{\omega} \sim \mu(\boldsymbol{x}; \boldsymbol{\omega}) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma_N^2)$. As usual, we assume that the noise is independent for different inputs $\boldsymbol{x}$ and parameters $\boldsymbol{\omega}$. Homoscedastic noise is a special case of the general heteroscedastic setting: the noise variance is simply a constant. Our approach could easily be extended to heteroscedastic noise by substituting a function $\sigma_N(\boldsymbol{x}; \boldsymbol{\omega})$ for σ_N, but for this work we limit ourselves to the simplest case.

10.2.1 Kernel-based Methods & Information Theory

We build on Holzmüller et al. [2022] which expresses contemporary batch active learning methods using kernel methods. Specifically, active learning is performed using kernel-based surrogates for deep neural network models. In this paper, we focus on surrogates using empirical covariance kernels instead of gradient kernels. While a full treatment of Holzmüller et al. [2022] is beyond the scope of this chapter, we briefly review some key ideas here. We refer the reader to the extensive paper for more details.

Gaussian Processes. Gaussian Processes are one way to introduce kernel-based methods. A simple way to think about Gaussian Processes [Williams and Rasmussen, 2006; Lázaro-Gredilla and Figueiras-Vidal, 2010; Rudner et al., 2022] is as a Bayesian linear regression model with an implicit, potentially infinite-dimensional feature space (depending on the covariance kernel) that uses the kernel trick to abstract away the feature map which maps the input space to the feature space.

Multivariate Gaussian Distribution. The distinctive property of a Gaussian Process is that all predictions are jointly Gaussian distributed. We can then write the joint distribution for a univariate regression model as:

$$Y_1, \ldots, Y_n \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \sim \mathcal{N}(\mathbf{0}, \operatorname{Cov}[\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_n)] + \sigma_N^2 \mathbf{I}), \tag{10.2}$$

where $\mu(\boldsymbol{x})$ are the observation-noise free predictions as random variables, and $\operatorname{Cov}[\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_n)]$ is the covariance matrix of the predictions. The covariance matrix is defined via the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$:

$$\operatorname{Cov}[\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_n)] = \left[k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right]_{i,j=1}^{n,n}. \tag{10.3}$$

The kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ can be chosen almost arbitrarily as long as it is positive semi-definite, e.g. see Williams and Rasmussen [2006, Ch. 4]. The linear kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x} \cdot \boldsymbol{x}'$ and the radial basis function kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{1}{2}|\boldsymbol{x} - \boldsymbol{x}'|^2)$ are common examples, as is the gradient kernel, which we examine next.

Fisher Information & Linearization. When using neural networks for regression, the gradient kernel

$$k_{\text{grad}}(\boldsymbol{x}, \boldsymbol{x}' \mid \boldsymbol{\omega}^*) \triangleq \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}; \boldsymbol{\omega}^*) \nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)]^{-1} \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}'; \boldsymbol{\omega}^*)^\top \tag{10.4}$$

$$= \langle \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}; \boldsymbol{\omega}^*), \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}'; \boldsymbol{\omega}^*) \rangle_{\nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)]^{-1}} \tag{10.5}$$

is the canonical choice, where $\boldsymbol{\omega}^*$ is a *maximum likelihood* or *maximum a posteriori estimate (MLE, MAP)* and $\nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)]$ is the Hessian of the prior at $\boldsymbol{\omega}^*$. Note that $\nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}; \boldsymbol{\omega}^*)$ is a *row* vector. Commonly, the prior is a Gaussian distribution with an identity covariance matrix, and thus $\nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)] = \mathbf{I}$.

The significance of this kernel lies in its relationship with the Fisher information matrix at $\boldsymbol{\omega}^*$ [Immer, 2020; Immer et al., 2021], as we have seen in §9, or equivalently, with the linearization of the loss function around $\boldsymbol{\omega}^*$ [Holzmüller et al., 2022]. This leads to a Gaussian approximation, which results in a Gaussian predictive posterior distribution when combined with a Gaussian likelihood. The use of the finite-dimensional gradient kernel thus results in an implicit Bayesian linear regression in the context of regression models.

Posterior Gradient Kernel. We can use the well-known properties of multivariate normal distributions to marginalize or condition the joint distribution in (10.2). Following Holzmüller et al. [2022], this allows us to explicitly obtain the posterior gradient kernel given additional $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ as:

$$k_{\text{grad}\to\text{post}(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)}(\boldsymbol{x}, \boldsymbol{x}' \mid \boldsymbol{\omega}^*) \tag{10.6}$$

$$\triangleq \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}; \boldsymbol{\omega}^*) \left(\sigma_N^{-2} \begin{pmatrix} \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}_1; \boldsymbol{\omega}^*) \\ \vdots \\ \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}_n; \boldsymbol{\omega}^*) \end{pmatrix} \begin{pmatrix} \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}_1; \boldsymbol{\omega}^*) \\ \vdots \\ \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}_n; \boldsymbol{\omega}^*) \end{pmatrix}^\top + \nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)] \right)^{-1} \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}'; \boldsymbol{\omega}^*)^\top.$$

The factor σ_N^{-2} originates from implicitly conditioning on $Y_i \mid \boldsymbol{x}_i$, which include observation noise.

Importantly for active learning, the multivariate normal distribution is the maximum entropy distribution for a given covariance matrix, and is thus an upper-bound for the entropy of any distribution with the same covariance matrix. The entropy is given by the log-determinant of the covariance matrix:

$$\mathrm{H}[Y_1, \ldots, Y_n \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] = \frac{1}{2} \log \det(\mathrm{Cov}[\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_n)] + \sigma^2 \mathbf{I}) + C_n, \tag{10.7}$$

where $C_n \triangleq \frac{n}{2} \log(2\,\pi\,e)$ is a constant that only depends on the number of samples n. Connecting kernel-based methods to information-theoretic quantities like the expected information gain, the respective acquisition scores are upper-bounds on the expected information gain, as we have examined in the previous chapter.

Expected Information Gain for Regression. In §4, for acquisition of samples $\boldsymbol{x}_{1..K}^{\text{acq}}$ for classification tasks, the EIG was defined via BatchBALD as:

$$\mathrm{I}[Y_{1..K}^{\text{acq}}; \boldsymbol{\Omega} \mid \boldsymbol{x}_{1..K}^{\text{acq}}, \mathcal{D}^{\text{train}}] = \mathrm{H}[Y_{1..K}^{\text{acq}} \mid \boldsymbol{x}_{1..K}^{\text{acq}}, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y_{1..K}^{\text{acq}} \mid \boldsymbol{x}_{1..K}^{\text{acq}}, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}]. \tag{4.2}$$

using the mutual information between the parameters $\boldsymbol{\omega}$ and the predictions $\{Y_i\}$ on an acquisition candidate batch $\{\boldsymbol{x}_i\}$. The conditional entropy term is precisely the entropy of a normal distribution given our model assumptions for the observation noise and can be computed exactly. In particular, for our regression task, as we assume fixed aleatoric uncertainty, the conditional entropy is constant in $\boldsymbol{x}$ and can even be dropped from the objective. BALD and the EIG for models with fixed aleatoric are equivalent to entropy maximization, which is a common objective in BOED [Hernández-Lobato et al., 2016]. To estimate BALD, we thus only need to compute the joint entropy, which can be upper-bounded by the corresponding entropy of the multivariate normal distribution with the same covariance matrix, yielding an upper-bound overall.

10.2.2 Black-Box Batch Active Learning

We more formally introduce the empirical covariance kernel and compare it to the gradient kernel commonly used for active learning with deep learning models in parameter-space. For non-differentiable models, we show how it can also be derived using a Bayesian model perspective on the hypothesis space.

In addition to being illustrative, this section allows us to connect prediction-based kernels to the kernels used by Holzmüller et al. [2022], which in turns connects them to various SotA active learning methods.

10.2.2.1 Predictive Covariance Kernel

To perform black-box batch active learning, we directly use the *predictive covariance* of $Y_i|\boldsymbol{x}_i$ and $Y_j|\boldsymbol{x}_j$:

$$\mathrm{Cov}_{\boldsymbol{\Omega}}[Y_i; Y_j \mid \boldsymbol{x}_i, \boldsymbol{x}_j] = \mathrm{Cov}_{\boldsymbol{\Omega}}[\mu_{\boldsymbol{x}_i}^{\boldsymbol{\omega}}; \mu_{\boldsymbol{x}_j}^{\boldsymbol{\omega}}] + \sigma_N^2 \mathbb{1}\{i = j\}, \tag{10.8}$$

where we have abbreviated $\mu(\boldsymbol{x}; \boldsymbol{\omega})$ with $\mu_{\boldsymbol{x}}^{\boldsymbol{\omega}}$, and used the law of total covariance and the fact that the noise is uncorrelated between samples.

We define the *predictive covariance kernel* $k_{\text{pred}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as the covariance of the predicted means:

$$k_{\text{pred}}(\boldsymbol{x}_i; \boldsymbol{x}_j) \triangleq \mathrm{Cov}_{\boldsymbol{\Omega}}[\mu_{\boldsymbol{x}_i}^{\boldsymbol{\omega}}; \mu_{\boldsymbol{x}_j}^{\boldsymbol{\omega}}]. \tag{10.9}$$

Compared to §10.2.1, we do not define the covariance via the kernel but the kernel via the covariance.

This is also simply known as the *covariance kernel* in the literature [Shawe-Taylor et al., 2004]. We use the prefix *predictive* to make clear that we look at the covariance of the predictions. The resulting Gram matrix is equal the covariance matrix of the predictions and positive definite (for positive σ_N and otherwise positive semi-definite), and thus the kernel is a valid kernel.

10.2.2.2 Empirical Predictive Covariance Kernel

For K sampled model parameters $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K \sim \mathrm{p}(\boldsymbol{\omega})$—for example, the members of a deep ensemble—the *empirical predictive covariance kernel* $k_{\widehat{\mathrm{pred}}}(\boldsymbol{x}_i; \boldsymbol{x}_j)$ is the empirical estimate:

$$k_{\widehat{\mathrm{pred}}}(\boldsymbol{x}_i; \boldsymbol{x}_j) \triangleq \widehat{\mathrm{Cov}}_{\boldsymbol{\Omega}}[\mu^{\boldsymbol{\omega}}_{\boldsymbol{x}_i}; \mu^{\boldsymbol{\omega}}_{\boldsymbol{x}_j}] = \frac{1}{K} \sum_{k=1}^{K} \left(\mu^{\boldsymbol{\omega}_k}_{\boldsymbol{x}_i} - \frac{1}{K} \sum_{l=1}^{K} \mu^{\boldsymbol{\omega}_l}_{\boldsymbol{x}_i} \right)^\top \left(\mu^{\boldsymbol{\omega}_k}_{\boldsymbol{x}_j} - \frac{1}{K} \sum_{l=1}^{K} \mu^{\boldsymbol{\omega}_l}_{\boldsymbol{x}_j} \right) \tag{10.10}$$

$$= \left\langle \frac{1}{\sqrt{K}} (\bar{\mu}^{\boldsymbol{\omega}_1}_{\boldsymbol{x}_i}, \ldots, \bar{\mu}^{\boldsymbol{\omega}_K}_{\boldsymbol{x}_i}), \frac{1}{\sqrt{K}} (\bar{\mu}^{\boldsymbol{\omega}_1}_{\boldsymbol{x}_j}, \ldots, \bar{\mu}^{\boldsymbol{\omega}_K}_{\boldsymbol{x}_j}) \right\rangle, \tag{10.11}$$

with centered predictions $\bar{\mu}^{\boldsymbol{\omega}_k}_{\boldsymbol{x}} \triangleq \mu^{\boldsymbol{\omega}_k}_{\boldsymbol{x}} - \frac{1}{K} \sum_{l=1}^{K} \mu^{\boldsymbol{\omega}_l}_{\boldsymbol{x}}$. As we can write this kernel as an inner product, it also immediately follows that the empirical predictive covariance kernel is a valid kernel and positive semi-definite.

10.2.2.3 Differentiable Models

Similar to Holzmüller et al. [2022, §C.1], we show that the posterior gradient kernel is a first-order approximation of the (predictive) covariance kernel. This section explicitly conditions on $\mathcal{D}^{\mathrm{train}}$. The result is simple but instructive:

Proposition 10.1. *The* posterior *gradient kernel* $k_{\mathrm{grad}\to\mathrm{post}(\mathcal{D}^{\mathrm{train}})}(\boldsymbol{x}_i; \boldsymbol{x}_j \mid \boldsymbol{\omega}^*)$ *is an approximation of the predictive covariance kernel* $k_{\mathrm{pred}}(\boldsymbol{x}_i; \boldsymbol{x}_j)$.

Proof. We use a first-order Taylor expansion of the mean function $\mu(\boldsymbol{x}; \boldsymbol{\omega})$ around $\boldsymbol{\omega}^*$:

$$\mu(\boldsymbol{x}; \boldsymbol{\omega}) \approx \mu(\boldsymbol{x}; \boldsymbol{\omega}^*) + \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}; \boldsymbol{\omega}^*) \underbrace{(\boldsymbol{\omega} - \boldsymbol{\omega}^*)}_{\triangleq \Delta\boldsymbol{\omega}}. \tag{10.12}$$

Choose $\boldsymbol{\omega}^* = \mathbb{E}_{\boldsymbol{\omega} \sim \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\mathrm{train}})}[\boldsymbol{\omega}]$ (BMA). Then we have $\mathbb{E}_{\mathrm{p}(w \mid \mathcal{D}^{\mathrm{train}})}[\mu(\boldsymbol{x}; \boldsymbol{\omega})] = \mu(\boldsymbol{x}; \boldsymbol{\omega}^*)$. We then have:

$$k_{\mathrm{pred}}(\boldsymbol{x}_i; \boldsymbol{x}_j) = \mathrm{Cov}_{\boldsymbol{\omega} \sim \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\mathrm{train}})}[\mu(\boldsymbol{x}_i; \boldsymbol{\omega}); \mu(\boldsymbol{x}_j; \boldsymbol{\omega})] \tag{10.13}$$

$$\approx \mathbb{E}_{\boldsymbol{\omega}^* + \Delta\boldsymbol{\omega} \sim \mathrm{p}(w \mid \mathcal{D}^{\mathrm{train}})}[\langle \nabla_{\boldsymbol{\omega}} \mu^{\boldsymbol{\omega}^*}_{\boldsymbol{x}_i} \Delta\boldsymbol{\omega}, \nabla_{\boldsymbol{\omega}} \mu^{\boldsymbol{\omega}^*}_{\boldsymbol{x}_j} \Delta\boldsymbol{\omega} \rangle] \tag{10.14}$$

$$= \nabla_{\boldsymbol{\omega}} \mu^{\boldsymbol{\omega}^*}_{\boldsymbol{x}_i} \; \mathbb{E}_{\boldsymbol{\omega}^* + \Delta\boldsymbol{\omega} \sim \mathrm{p}(w \mid \mathcal{D}^{\mathrm{train}})}[\Delta\boldsymbol{\omega} \Delta\boldsymbol{\omega}^\top] \, \nabla_{\boldsymbol{\omega}} \mu^{\boldsymbol{\omega}^*}_{\boldsymbol{x}_j}{}^\top \tag{10.15}$$

$$= \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}_i; \boldsymbol{\omega}^*) \; \mathrm{Cov}[\boldsymbol{\Omega} \mid \mathcal{D}^{\mathrm{train}}] \, \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}_j; \boldsymbol{\omega}^*)^\top \tag{10.16}$$

$$\approx k_{\mathrm{grad}\to\mathrm{post}(\mathcal{D}^{\mathrm{train}})}(\boldsymbol{x}_i; \boldsymbol{x}_j \mid \boldsymbol{\omega}^*). \tag{10.17}$$

The intermediate expectation is the model covariance $\mathrm{Cov}[\boldsymbol{\Omega} \mid \mathcal{D}^{\mathrm{train}}]$ as $\boldsymbol{\omega}^*$ is the BMA. For the last step, we use the Gauss-Newton approximation again [Immer et al., 2021] and approximate the inverse of the covariance using the Hessian of the negative log likelihood at $\boldsymbol{\omega}^*$:

$$\mathrm{Cov}[\boldsymbol{\Omega} \mid \mathcal{D}^{\mathrm{train}}]^{-1} \approx \nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^* \mid \mathcal{D}^{\mathrm{train}})] \tag{10.18}$$

$$= \nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\mathcal{D}^{\mathrm{train}} \mid \boldsymbol{\omega}^*) - \log \mathrm{p}(\boldsymbol{\omega}^*)] \tag{10.19}$$

$$= \sigma^{-2} \textstyle\sum_i \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}^{\mathrm{train}}_i; \boldsymbol{\omega}^*)^\top \nabla_{\boldsymbol{\omega}} \mu(\boldsymbol{x}^{\mathrm{train}}_i; \boldsymbol{\omega}^*) \tag{10.20}$$

$$- \nabla^2_{\boldsymbol{\omega}} \log \mathrm{p}(\boldsymbol{\omega}^*), \tag{10.21}$$

where we have first used Bayes' theorem and that $\mathrm{p}(\mathcal{D}^{\text{train}})$ vanishes under differentiation—it is constant in $\boldsymbol{\omega}$. Secondly, the Hessian of the negative log likelihood is just the outer product of the gradients divided by the noise variance in the homoscedastic regression case. $\nabla^2_{\boldsymbol{\omega}}[-\log \mathrm{p}(\boldsymbol{\omega}^*)]$ is the prior term. This matches (10.6). □

10.2.2.4 Non-Differentiable Models

How can we apply the above result to non-differentiable models? In the following, we use a Bayesian view on the hypothesis space to show that we can connect the empirical predictive covariance kernel to a gradient kernel here, too. With $\hat{\boldsymbol{\Omega}} \triangleq (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K)$ fixed—e.g. these could be the different parameters of the members of a deep ensemble—we introduce a latent Ψ to represent the 'true' hypothesis $\boldsymbol{\omega}_\psi \in \hat{\boldsymbol{\Omega}}$ from this empirical hypothesis space $\hat{\boldsymbol{\Omega}}$, which we want to identify. This is similar to QbC [Seung et al., 1992]. In essence, the latent Ψ takes on the role of $\boldsymbol{\Omega}$ from the previous section, and we are interested in learning the 'true' Ψ from additional data. We, thus, examine the kernels for Ψ, as opposed to $\boldsymbol{\Omega}$.

Specifically, we model Ψ using a one-hot categorical distribution, that is a multinomial distribution from which we draw one sample: $\Psi \sim \text{Multinomial}(\boldsymbol{q}, 1)$, with $\boldsymbol{q} \in S^{K-1}$ parameterizing the distribution, where S^{K-1} denotes the $K-1$ simplex in $\mathbb{R}^K$. Then, $\boldsymbol{q}_k = \mathrm{p}(\Psi = e_k)$, where e_k denotes the k-th unit vector; and $\sum_{k=1}^K \boldsymbol{q}_k = 1$. For the predictive $\tilde{\mu}(\boldsymbol{x}; \Psi)$, we have:

$$\tilde{\mu}(\boldsymbol{x}; \Psi) \triangleq \mu(\boldsymbol{x}; \boldsymbol{\omega}_\Psi) = \langle \mu(\boldsymbol{x}; \cdot), \Psi \rangle, \tag{10.22}$$

where we use $\boldsymbol{\omega}_\psi$ to denote the $\boldsymbol{\omega}_k$ when we have $\psi = e_k$ in slight abuse of notation, and $\mu(\boldsymbol{x}; \cdot) \in \mathbb{R}^K$ is a column vector of the predictions $\mu(\boldsymbol{x}; \boldsymbol{\omega}_k)$ for $\boldsymbol{x}$ for all $\boldsymbol{\omega}_k$. This follows from ψ being a one-hot vector.

We now examine this model and its kernels. The BMA of $\tilde{\mu}(\boldsymbol{x}; \Psi)$ matches the previous empirical mean, and, if we choose $\boldsymbol{q}$ to have an uninformative[2] uniform distribution over the hypotheses ($\boldsymbol{q}_k \triangleq \frac{1}{K}$), we obtain:

$$\tilde{\mu}(\boldsymbol{x}; \boldsymbol{q}) \triangleq \mathbb{E}_{\mathrm{p}(\psi)}[\mu(\boldsymbol{x}; \boldsymbol{\omega}_\psi)] = \langle \mu(\boldsymbol{x}; \cdot), \boldsymbol{q} \rangle = \sum_{\psi=1}^K \boldsymbol{q}_\psi \mu(\boldsymbol{x}; \boldsymbol{\omega}_\psi) = \sum_{\psi=1}^K \frac{1}{K} \mu(\boldsymbol{x}; \boldsymbol{\omega}_\psi). \tag{10.23}$$

What is the predictive covariance kernel of this model? And what is the posterior gradient kernel for $\boldsymbol{q}$?

Proposition 10.2.

1. *The predictive covariance kernel $k_{\text{pred},\psi}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $\hat{\boldsymbol{\Omega}}$ using uniform $\boldsymbol{q}$ is equal to the empirical predictive covariance kernel $k_{\widehat{\text{pred}}}(\boldsymbol{x}_i; \boldsymbol{x}_j)$.*
2. *The 'posterior' gradient kernel $k_{\text{grad},\psi \to \text{post}(\mathcal{D}^{\text{train}})}(\boldsymbol{x}_i; \boldsymbol{x}_j)$ for $\hat{\boldsymbol{\Omega}}$ in respect to Ψ using uniform $\boldsymbol{q}$ is equal to the empirical predictive covariance kernel $k_{\widehat{\text{pred}}}(\boldsymbol{x}_i; \boldsymbol{x}_j)$.*

Proof. Like for the previous differentiable model, the BMA of the model parameters Ψ is just $\boldsymbol{q}$: $\mathbb{E}[\Psi] = \boldsymbol{q}$. The first statement immediately follows:

$$k_{\text{pred},\psi}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{Cov}_\psi[\tilde{\mu}(\boldsymbol{x}_i; \psi); \tilde{\mu}(\boldsymbol{x}_j; \psi)] = \mathbb{E}_{\mathrm{p}(\psi)}[\bar{\mu}^{\boldsymbol{\omega}_\psi}_{\boldsymbol{x}_i} \bar{\mu}^{\boldsymbol{\omega}_\psi}_{\boldsymbol{x}_j}] \tag{10.24}$$

[2] If we had additional information about the $\hat{\boldsymbol{\Omega}}$—for example, if we had validation losses—we could use that to inform $\boldsymbol{q}$.

$$= \tfrac{1}{K} \sum_{\psi} \bar{\mu}^{\boldsymbol{\omega}_\psi}_{\boldsymbol{x}_i} \bar{\mu}^{\boldsymbol{\omega}_\psi}_{\boldsymbol{x}_j} = k_{\widehat{\text{pred}}}(\boldsymbol{x}_i; \boldsymbol{x}_j). \tag{10.25}$$

For the second statement, we will show that we can express the predictive covariance kernel as a linearization around Ψ. We can read off a linearization for $\nabla_\psi \tilde{\mu}(\boldsymbol{x}_i; \psi)$ from the inner product in Equation 10.23:

$$\nabla_\psi \tilde{\mu}(\boldsymbol{x}_i; \psi) = \mu(\boldsymbol{x}; \cdot)^\top, \tag{10.26}$$

This allows us to write the predictive covariance kernel as a linearization around $\boldsymbol{q}$:

$$\begin{aligned}
k_{\text{pred},\psi}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \text{Cov}_{\psi \sim \text{p}(\psi)}[\tilde{\mu}(\boldsymbol{x}_i; \psi); \tilde{\mu}(\boldsymbol{x}_j; \psi)] && (10.27)\\
&= \mathbb{E}_{\boldsymbol{q}+\Delta\psi \sim \text{p}(\psi)}[\nabla_\psi \tilde{\mu}(\boldsymbol{x}_i; \psi)\Delta\psi, \nabla_\psi \tilde{\mu}(\boldsymbol{x}_j; \psi)\Delta\psi] && (10.28)\\
&= \nabla_\psi \tilde{\mu}(\boldsymbol{x}_i; \boldsymbol{q})\ \text{Cov}[\Psi]\ \nabla_\psi \tilde{\mu}(\boldsymbol{x}_i; \boldsymbol{q})^\top && (10.29)\\
&= k_{\text{grad},\psi \to \text{post}(\mathcal{D}^{\text{train}})}(\boldsymbol{x}_i; \boldsymbol{x}_j). && (10.30)
\end{aligned}$$

□

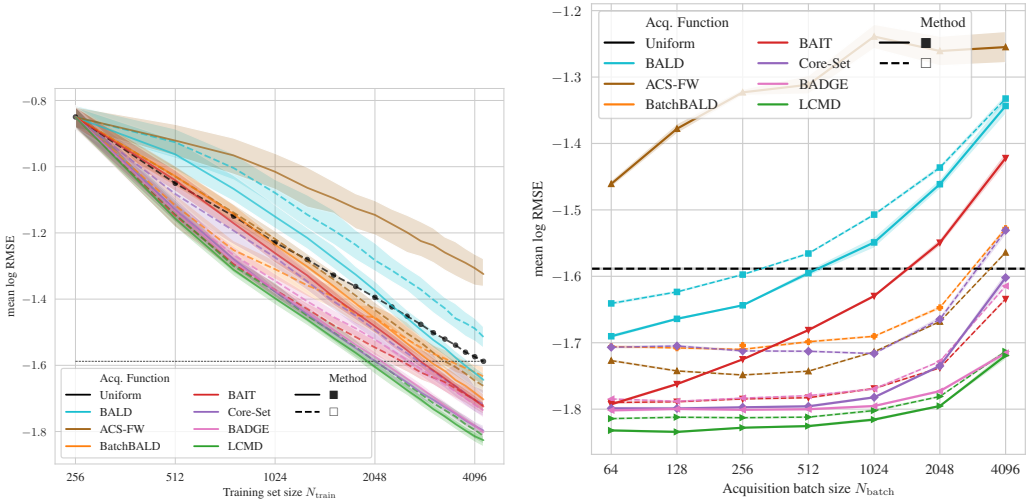
The above gradient kernel is only the posterior gradient kernel in the sense that we have sampled $\boldsymbol{\omega}_\psi$ from the non-differentiable model after inference on training data. The samples themselves are drawn uniformly.

The covariance of the multinomial Ψ is: $\text{Cov}[\Psi] = \text{diag}(\boldsymbol{q}) - \boldsymbol{q}\boldsymbol{q}^\top$. Thus, substituting, we can verify that the posterior gradient kernel is indeed equal to the predictive covariance kernel:

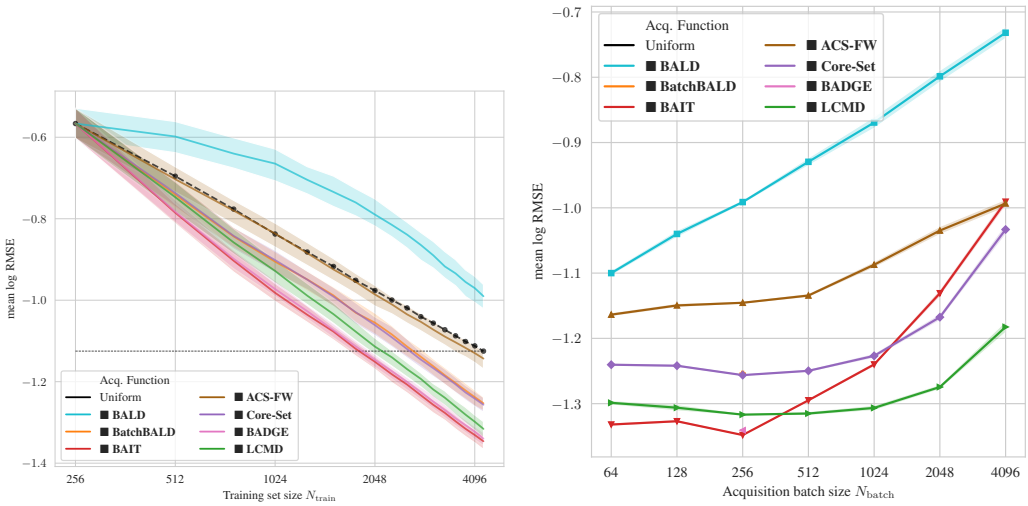
$$\begin{aligned}
k_{\text{grad},\psi \to \text{post}(\mathcal{D}^{\text{train}})}(\boldsymbol{x}_i; \boldsymbol{x}_j) &= \nabla_\psi \tilde{\mu}(\boldsymbol{x}_i; \boldsymbol{q})\, (\text{diag}(\boldsymbol{q}) - \boldsymbol{q}\boldsymbol{q}^\top)\, \nabla_\psi \tilde{\mu}(\boldsymbol{x}_i; \boldsymbol{q})^\top && (10.31)\\
&= \mu(\boldsymbol{x}_i; \cdot)^\top\, \text{diag}(\boldsymbol{q})\, \mu(\boldsymbol{x}_j; \cdot) - (\mu(\boldsymbol{x}_i; \cdot)^\top\, \boldsymbol{q})\, (\boldsymbol{q}^\top\, \mu(\boldsymbol{x}_j; \cdot)) && (10.32)\\
&= \frac{1}{K} \sum_{\psi} \mu(\boldsymbol{x}_i; \boldsymbol{\omega}_\psi)\, \mu(\boldsymbol{x}_j; \boldsymbol{\omega}_\psi)^\top && (10.33)\\
&\quad - \left(\frac{1}{K} \sum_{\psi} \mu(\boldsymbol{x}_i; \boldsymbol{\omega}_\psi) \right) \left(\frac{1}{K} \sum_{\psi} \mu(\boldsymbol{x}_j; \boldsymbol{\omega}_\psi) \right)\\
&= k_{\widehat{\text{pred}}}(\boldsymbol{x}_i; \boldsymbol{x}_j). && (10.34)
\end{aligned}$$

This demonstrates that a straightforward Bayesian model can be constructed on top of a non-differentiable ensemble model. Bayesian inference in this context aims to identify the most suitable member of the ensemble. Given the limited number of samples and likelihood of model misspecification, it is likely that none of the members accurately represents the true model. However, for active learning purposes, the main focus is solely on quantifying the degree of disagreement among the ensemble members.

A similar Bayesian model using Bayesian Model Combination (BMC) could be set up which allows for arbitrary convex mixtures of the ensemble members. This would entail using a Dirichlet distribution $\Psi \sim \text{Dirichlet}(\boldsymbol{\alpha})$ instead of the multinomial distribution. Assuming an uninformative prior ($\boldsymbol{\alpha}_k \triangleq 1/K$), this leads to the same results up to a constant factor of $1 + \sum_k \boldsymbol{\alpha}_k = 2$. This is pleasing because it does not matter whether we use a multinomial or Dirichlet distribution, that is: whether we take a hypothesis space view with a 'true' hypothesis or accept that our model is likely misspecified, and we are dealing with a mixture of models, the results are the same up to a constant factor.



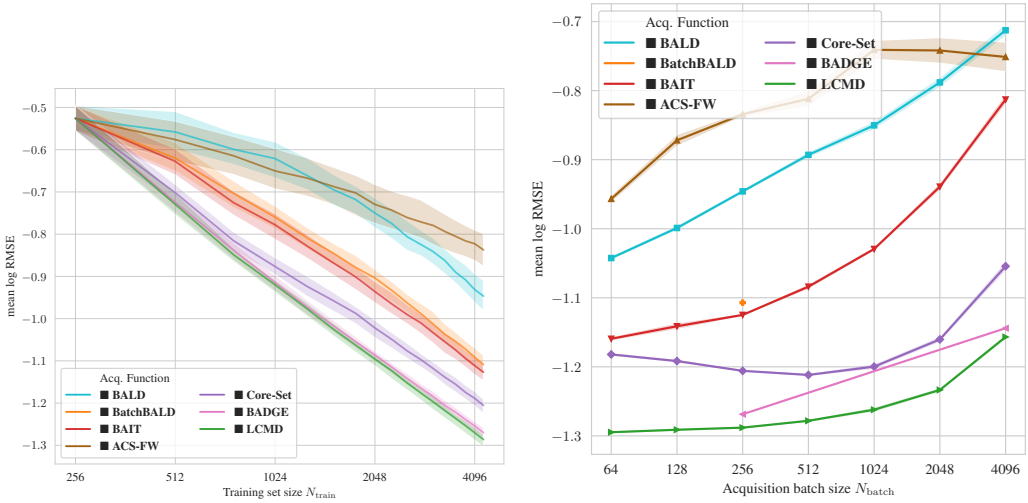
(a) Deep Neural Networks.



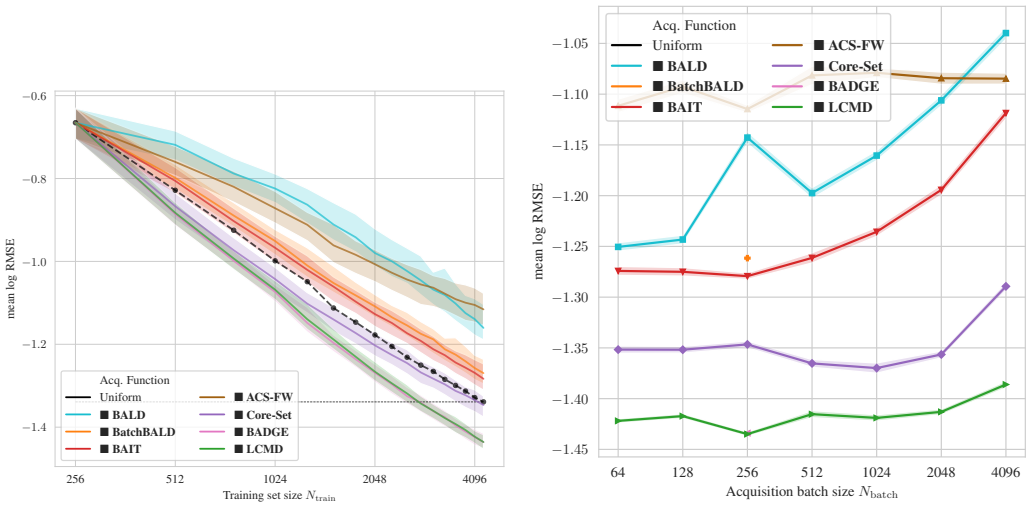
(b) Random Forests.

Figure 10.1: *Mean logarithmic RMSE over 15 regression datasets.* **((a))** For DNNs, we see that black-box ■ methods work as well as white-box □ methods, and in most cases better, except for ACS-FW and BAIT. **((b))** For random forests (100 estimators) with the default hyperparameters from scikit-learn [Pedregosa et al., 2011a], we see that black-box methods perform better than the uniform baseline, except for BALD, which uses top-K acquisition. In the appendix, see Table I.1 for average performance metrics and Figure I.2 and I.3 for plots with additional error metrics. Averaged over 20 trials.

Application to DNNs, BNNs, and Other Models. The proposed approach has relevance due to its versatility, as it can be applied to a wide range of models that can be consistently queried for prediction, including deep ensembles [Lakshminarayanan et al., 2017], Bayesian neural networks (BNNs) [Blundell et al., 2015; Gal and Ghahramani, 2016a], and non-differentiable models. The kernel used in this approach is simple to implement and scales in the number of empirical predictions per sample, rather than in the parameter space, as seen in other methods such as Ash et al. [2021].



(a) Random Forests (Bagging).



(b) Gradient-Boosted Trees.

Figure 10.2: *Mean logarithmic RMSE over 15 regression datasets (cont'd).* For random forests using bagging **((a))** with 10 bootstrapped training sets, and for gradient-boosted trees [Dorogush et al., 2018] **((b))** with a *virtual ensemble* of 20 members, we see that only a few of the black-box methods perform better than the uniform baseline: LCMD, BADGE and CoreSet. We hypothesize that the virtual ensembles and a bagged ensemble of random forests do not express as much predictive disagreement which leads to worse performance for active learning. In the appendix, see Table I.1 for average performance metrics and Figure I.4 and I.5 for plots with additional error metrics. Averaged over 20 trials.

10.3 Empirical Validation

We follow the evaluation from Holzmüller et al. [2022] and use their framework to ease comparison. This allows us to directly compare to several SotA methods in a regression setting, respectively their kernel-based analogues. Specifically, we compare to the following popular deep active learning methods: BALD [Houlsby et al., 2011], BatchBALD (§4), BAIT [Ash et al., 2021], BADGE [Ash et al., 2020], ACS-FW [Pinsler et al., 2019], Core-Set [Sener and Savarese, 2018]/FF-Active [Geifman

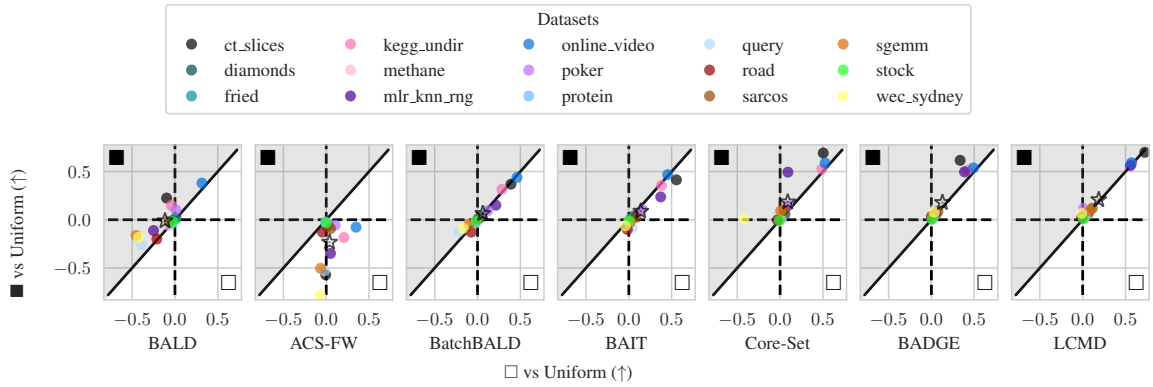


Figure 10.3: *Average Logarithmic RMSE by regression datasets for DNNs:* ■ *vs* □ *(vs Uniform).* Across acquisition functions, the performance of black-box methods is highly correlated with the performance of white-box methods, even though black-box methods make fewer assumptions about the model. We plot the improvement of the white-box □ method over the uniform baseline on the x-axis (so for datasets with markers right of the dashed vertical lines, the white-box method performs better than uniform) and the improvement of the black-box ■ method over the uniform baseline on the y-axis (so for datasets with markers above the dashed horizontal lines, the black-box method performs better than uniform). For datasets with markers in the ■ *diagonal half*, the black-box method performs better than the white-box method. The average over all datasets is marked with a star ⋆. Surprisingly, on average over all acquisition rounds, the black-box methods perform slightly better than the white-box methods for all but ACS-FW and BAIT. In the appendix, see Figure I.1 for the final acquisition round and Table I.3 for details on the datasets. Averaged over 20 trials.

and El-Yaniv, 2017], and LCMD [Holzmüller et al., 2022]. We also compare to the random selection baseline ('Uniform'). We use 15 large tabular datasets from the UCI Machine Learning Repository [Dua and Graff, 2017] and the OpenML benchmark suite [Vanschoren et al., 2013] for our experiments: *sgemm* (SGEMM GPU kernel performance); *wec_sydney* (Wave Energy Converters); *ct_slices* (Relative location of CT slices on axial axis); *kegg_undir* (KEGG Metabolic Reaction Network - Undirected); *online_video* (Online Video Characteristics and Transcoding Time); *query* (Query Analytics Workloads); *poker* (Poker Hand); *road* (3D Road Network - North Jutland, Denmark); *mlr_knn_rng*; *fried*; *diamonds*; *methane*; *stock* (BNG stock); *protein* (physicochemical-protein); *sarcos* (SARCOS data). See Table I.3 in the appendix for more details.

Experimental Setup. We use the same experimental setup and hyperparameters as Holzmüller et al. [2022]. We report the logarithmic RMSE averaged over 20 trials for each dataset and method. For ensembles, we compute the performance for each ensemble member separately, enabling a fair comparison to the non-ensemble methods. Performance differences can thus be attributed to the acquisition function, rather than the ensemble. We used A100 GPUs with 40 GB of GPU memory.

Ensemble Size. For deep learning, we use a small ensemble of 10 models, which is sufficient to achieve good performance. This ensemble size can be considered 'small' in the regression setting [Lázaro-Gredilla and Figueiras-Vidal, 2010; Zhang and Zhou, 2013], whereas in Bayesian Optimal Experiment Design much higher sample counts are regularly used [Foster et al., 2021]. In many cases, training an ensemble of regression

models is still fast and could be considered cheap compared to the cost of acquiring additional labels. For non-differentiable models, we experiment with random forests [Breiman, 2001], using the different trees as ensemble members, and a virtual ensemble of gradient-boosted decision trees [Prokhorenkova et al., 2017]. For random forests, we use the implementation provided in scikit-learn [Pedregosa et al., 2011a] with default hyperparameters, that is using 100 trees per forest. We use the predictions from each tree as a virtual ensemble member. We do not perform any hyperparameter tuning. We also report results for random forests with bagging, where we train a real ensemble of 10 random forests. For gradient-boosted decision trees, we use a virtual ensemble of up to 20 members with early stopping using a validation set[3]. We use the implementation in CatBoost [Dorogush et al., 2018]. We do not perform any hyperparameter tuning.

Black-Box vs White-Box Deep Active Learning. In Figure 10.1(a) and 10.3, we see that B³AL is competitive with white-box active learning, when using BALD, BatchBALD, BAIT, BADGE, and Core-Set. On average, B³AL outperforms the white-box methods on the 15 datasets we analyzed (excluding ACS-FW and BAIT). We hypothesize that this is due to the implicit Fisher information approximation in the white-box methods we have examined (§9), which is not as accurate in the low data regime as the more explicit approximation in B³AL via ensembling. On the other hand, it seems that the black-box methods suffer from the low feature space dimensionality, as they are much closer to BALD.

Why can Black-Box Methods Outperform White-Box Methods? Following §10.2, white-box and black-box methods are based on kernels, which can be seen as different *approximations* of the predictive covariance kernel. White-box methods implicitly assume that the predictive covariance kernel is well approximated by the Fisher information kernel and the gradient kernel (§9). However, Long [2022] demonstrated that this assumption does not always hold, particularly in low data regimes, where a Gaussian might not approximate the parameter distribution well. Instead, Long [2022] suggests using a multimodal distribution. In these situations, methods that employ ensembling, such as B³AL, to approximate the predictive covariance kernel are more robust. The different ensemble members can reside in different modes of the parameter distribution, allowing black-box methods to outperform their white-box counterparts.

Non-Differentiable Models. In Figure 10.1(b), 10.2(a), and 10.2(b), we observe that B³AL is effective for non-differentiable models, including random forests and gradient-boosted decision trees. BALD for non-differentiable models can be considered equivalent to QbC [Seung et al., 1992], while BatchBALD for non-differentiable models can be viewed as equivalent to QbC with batch acquisition [Nguyen et al., 2012]. For random forests, all methods except BALD (using top-k selection) outperform uniform acquisition. However, for random forests with bagging and gradient-boosted decision trees, B³AL surpasses random acquisition only when employing LMCD and BADGE. This may be attributed to the reduced disagreement within a virtual ensemble for gradient-boosted decision trees and between distinct random forests. In particular, random forests with bagging appear to support this explanation, as a single random forest seems to exhibit more disagreement among its individual trees than an ensemble

[3]If the virtual ensemble creation fails because there are no sufficiently many trees due to early stopping, we halve the ensemble size and retry. This was only needed for the *poker* dataset.

of random forests with bagging does between different forests. This is evident in the superior overall active learning performance of the single random forest compared to the ensemble of random forests with bagging, c.f. Figure 10.1(b) and 10.2(a).

10.4 Discussion

In this chapter, we have demonstrated the effectiveness of a simple extension to kernel-based methods that utilizes empirical predictions rather than gradient kernels. This modification enables black-box batch active learning with good performance. Importantly, B³AL also generalizes to non-differentiable models, an area that has received limited attention as of late.

This result also partially answer one of the research questions from the previous chapter (§9): how do prediction-based methods compare to parameter-based ones? We find that for regression the prediction-based methods are competitive with the parameter-based methods in batch active learning.

The main limitation of our proposed approach lies in the acquisition of a sufficient amount of empirical predictions. This could be a challenge, particularly when using deep ensembles with larger models or non-differentiable models that cannot be parallelized efficiently. Our experiments using virtual ensembles indicate that the diversity of the ensemble members plays a crucial role in determining the performance.

Likewise, the main limitation of this chapter and the empirical comparisons is that we only consider regression tasks. Extending the results to classification is an important direction for future work.

What gets sharper the more you use it?

11 Conclusion

In this thesis, we set out to address several research questions focused on active learning and data subset selection in deep learning, with an emphasis on information-theory intuitions:

1. How can uncertainty quantification, specifically aleatoric and epistemic uncertainty, be better understood and applied in the context of active learning and active sampling?
2. How can a deeper understanding of the theoretical foundations of active learning and active sampling contribute to the progress of the field and improve the practical application of these techniques?
3. How can the cost of gathering and labeling data be reduced, and how can training be sped up in a principled fashion?
4. What are the connections between different active learning and active sampling approaches, and how can information theory be used to unify these approaches?

Our main findings and contributions include the unifying perspective provided by information-theory intuitions, the discovery that many existing methods can be explained using the same framework, and the effectiveness of simple methods in various applications.

Regarding the first research question, we have demonstrated that by examining aleatoric and epistemic uncertainty in more detail, we can propose new baselines for uncertainty quantification using single forward-pass deep neural networks (§3). This approach allows us to quantify epistemic uncertainty well and achieve competitive results in active learning without having to be Bayesian.

In addressing the second research question, this thesis contributes to a deeper understanding of the theoretical foundations of active learning and active sampling by providing a unified framework based on information theory (§2 and §9). This framework helps researchers better understand the trade-offs and connections between different approaches, ultimately improving the practical application of these techniques.

To answer the third research question, we have explored various methods for reducing the cost of gathering and labeling data (§3 to 5, §7 and §10) and speeding up training in a principled fashion (§8 and Appendix B.1). While we have provided a principled approach for batch acquisition in §4, it is too slow at larger acquisition batch sizes and can suffer from issues related to the estimation of joint predictives for larger batch acquisition sizes. On the other hand, our conceptually simpler and effective methods, such as stochastic batch acquisition (§5) and single forward-pass deterministic methods (§3), have shown promising results and have the potential to benefit practitioners in a wide range of applications.

Finally, in addressing the fourth research question, we have identified connections between different active learning and active sampling approaches by using information theory to unify these methods (§9 and §10). This unifying perspective allows us to better understand the relationships between various techniques and access the principled framework for reasoning about uncertainty and informativeness that information theory provides.

Nonetheless, there are some obvious limitations to our research. One of the main challenges is improving joint predictions for multiple samples (§6). Additionally, a more satisfying approach for classification tasks and beyond is needed when using kernel-based methods. Finally, there are various related areas such as active testing, active inference, and exploration in reinforcement learning that are yet to be connected in more detail to active learning and active sampling.

Future research directions could include developing more effective methods for joint predictions and batch selection, exploring other alternative approaches for classification, and investigating the quality of the chosen approximations. We detail several other specific research questions in various chapters: in particular in §6 and §9.

In conclusion, this thesis has advanced our understanding of active learning and data subset selection in deep learning by providing a unifying perspective based on information-theory intuitions. We hope this thesis has also contributed to the reader's understanding and that our contributions will inspire further research and help practitioners adopt effective techniques in their work.

Appendices

Can you imagine what I would do if I could do all I can?

Sun Tzu, The Art of War

A

Causal-BALD: Deep Bayesian Active Learning of Outcomes to Infer Treatment-Effects from Observational Data

How will a patient's health be affected by taking a medication [Perez, 2019]? How will a user's question be answered by a search recommendation [Noble, 2018]? Insight into these questions can be gained by learning about personalized treatment effects. Estimating personalized treatment effects from observational data is essential in situations where experimental designs are infeasible, unethical or expensive. Observational data represent a population of individuals described by a set of pre-treatment covariates (age, blood pressure, socioeconomic status), an assigned treatment (medication, no medication), and a post-treatment outcome (severity of migraines). An ideal personalized treatment effect is the difference between the post-treatment outcome had the individual been treated, and the outcome had they not been treated. But, it is impossible to observe both outcomes for an individual, so the difference must instead be computed between populations. Therefore, in the common setting of binary treatments, data is partitioned into the *treatment group* (individuals that received the treatment) and the *control group* (individuals who did not). The personalized treatment effect is then given by the expected difference in outcomes between treated and controlled individuals who share the same (or similar) measured covariates (difference between solid lines in the middle pane of Figure A.1).

Increasingly, pre-treatment covariates are being assembled from high-dimensional, heterogeneous measurements such as medical images and electronic health records [Sudlow et al., 2015]. Deep learning methods have been shown capable of learning personalized treatment effects from such data [Shalit et al., 2017; Shi et al., 2019; Jesson et al., 2021]. However, a key problem in deep learning is data efficiency. While modern methods are capable of impressive performance, they need a significant amount of labeled data. Acquiring labeled data can be expensive, often requiring specialist knowledge or an invasive procedure to determine the outcome. Therefore, it is desirable to minimize the amount of labeled data needed to obtain a well-performing model. Active learning provides a principled framework to address this concern [Cohn et al., 1996]. In active learning for treatment effects [Deng et al., 2011; Sundin et al., 2019], a model is trained on available labeled data consisting of covariates, assigned treatments, and acquired outcomes. The model predictions are then used to select the most informative examples from a set of data consisting of only covariates and treatment indicators. Outcomes are then acquired, e.g. by performing a biopsy, for the selected

patients and the model is retrained and evaluated. This process is repeated until either a satisfactory performance level is achieved, or the labeling budget is exhausted.

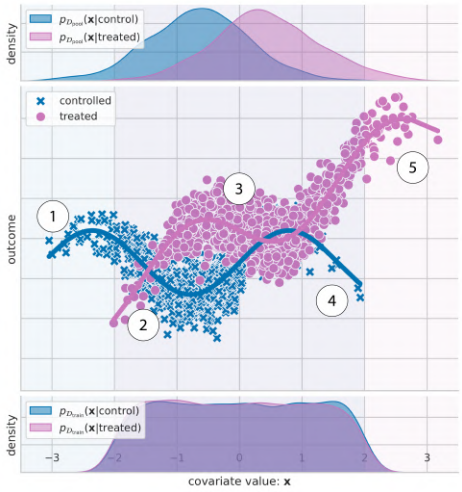


Figure A.1: Observational data. Top: data density of treatment (right) and control (left) groups. Middle: observed outcome response for treatment (circles) and control (x's) groups. Bottom: data density for active learned training set after a number of acquisition steps.

At first sight this might seem simple; however, active learning induces bias resulting in divergence between the distribution of the acquired training data and the distribution of the pool set data [Farquhar et al., 2021]. In the context of learning causal effects, such bias has important positive and negative consequences. For example, while random acquisition active learning results in an unbiased sample of the training data, it can lead to over-allocation of resources to the mode of the data at the expense of learning about underrepresented data. Conversely, while biasing acquisitions toward lower density regions of the pool data can be desirable, it can also lead to acquisitions for which we cannot know the treatment effect, which could in turn lead to uninformed, potentially harmful, personalized decisions.

To gain insight into how biasing the acquisition of training data can be beneficial for learning treatment effects, consider a key difference between experimental and observational data: the treatment assignment mechanism is not available for observational data. This means that there may be unobserved variables that affect treatment assignment (an untestable condition), but also that the relative proportion of individuals treated to those controlled can vary across different subpopulations of the data. The later point is illustrated in Figure A.1, where there are relatively equal proportions of treated and controlled examples for data in region 3, but the proportions become less balanced as we move to either the left or the right. In extreme cases, say if a group described by some covariate values were systematically excluded from treatment, the treatment effect for that group *cannot be known* [Petersen et al., 2012]. This is illustrated in Figure A.1 by region 1, where there are only controlled examples, and by region 5, where there are only treated examples. In the language of causal inference, the necessity of seeing both treated and untreated examples for each subpopulation corresponds to satisfaction of the overlap (or positivity) assumption (see A.3). Regions 2 and 4 of Figure A.1 are interesting as while either the treated or control group are underrepresented, there may still be sufficient coverage to learn treatment effects.

We propose that the acquisition of unlabeled data should focus on exploring all regions with sufficient overlap, but not areas with no overlap. The bottom pane of Figure A.1 imagines what a resulting training set distribution could look like at an intermediate active learning step. It is not trivial to design such acquisition functions: naively applying active learning acquisition functions results in suboptimal and sample inefficient acquisitions of training examples, as we show below. To this end, we develop an epistemic uncertainty aware method for active learning of personalized treatment effects from high dimensional observational data. We demonstrate the performance of the proposed acquisition strategies on a synthetic and semisynthetic datasets.

A.1 Background

A.1.1 Estimation of Personalized Treatment-Effects

Personalized treatment-effect estimation seeks to know the effect of a treatment $\mathrm{T} \in \mathcal{T}$ on the outcome $Y \in \mathcal{Y}$ for individuals described by covariates $\boldsymbol{X} \in \mathcal{X}$. In this chapter, we consider the random variable (r.v.) T to be binary ($\mathcal{T} = \{0, 1\}$), the r.v. Y to be part of a bounded set $\mathcal{Y}$, and $\boldsymbol{X}$ to be a multi-variate r.v. of dimension d ($\mathcal{X} = \mathbb{R}^d$). Under the Neyman-Rubin causal model [Neyman, 1923; Rubin, 1974], the individual treatment effect (ITE) for a person u is defined as the difference in potential outcomes $Y^1(u) - Y^0(u)$, where the r.v. Y^1 represents the potential outcome were they *treated*, and the r.v. Y^0 represents the potential outcome were they *controlled* (not treated). Realizations of the random variables $\boldsymbol{X}$, T, Y, Y^0, and Y^1 are denoted by $\boldsymbol{x}$, t, y, y^0, and y^1, respectively.

The ITE is a fundamentally unidentifiable quantity, so instead we look at the expected difference in potential outcomes for individuals described by $\boldsymbol{X}$, or the Conditional Average Treatment Effect (CATE): $\tau(\boldsymbol{x}) \equiv \mathbb{E}[Y^1 - Y^0 \mid \boldsymbol{X} = \boldsymbol{x}]$ Abrevaya et al. [2015]. The CATE is identifiable from an observational dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \mathrm{t}_i, y_i)\}_{i=1}^n$ of samples $(\boldsymbol{x}_i, \mathrm{t}_i, y_i)$ from the joint empirical distribution $P_{\mathcal{D}}(\boldsymbol{X}, \mathrm{T}, Y^0, Y^1)$, under the following three assumptions:

Assumption A.1. (Consistency) $y = \mathrm{t}y^{\mathrm{t}} + (1 - \mathrm{t})y^{1-\mathrm{t}}$, i.e. an individual's observed outcome y given assigned treatment t is identical to their potential outcome y^{t}.

Assumption A.2. (Unconfoundedness) $(Y^0, Y^1) \perp\!\!\!\perp \mathrm{T} \mid \boldsymbol{X}$.

Assumption A.3. (Overlap) $0 < \pi_{\mathrm{t}}(\boldsymbol{x}) < 1 : \forall \mathrm{t} \in \mathcal{T}$,

where $\pi_{\mathrm{t}}(\boldsymbol{x}) \equiv \mathrm{P}(\mathrm{T} = \mathrm{t} \mid \boldsymbol{X} = \boldsymbol{x})$ is the **propensity for treatment** for individuals described by covariates $\boldsymbol{X} = \boldsymbol{x}$ Rubin [1974]. When these assumptions are satisfied, $\widehat{\tau}(\boldsymbol{x}) \equiv \mathbb{E}[Y \mid \mathrm{T} = 1, \boldsymbol{X} = \boldsymbol{x}] - \mathbb{E}[Y \mid \mathrm{T} = 0, \boldsymbol{X} = \boldsymbol{x}]$ is an identifiable, unbiased estimator of $\tau(\boldsymbol{x})$.

A variety of parametric [Robins et al., 2000; Tian et al., 2014; Shalit et al., 2017] and non-parametric estimators [Hill, 2011; Xie et al., 2012; ala, 2017; Gao and Han, 2020] have been proposed for CATE. Here, we focus on parametric estimators for compactness. Parametric CATE estimators assume that outcomes y are generated according to a likelihood $p_{\boldsymbol{\omega}}(y \mid \boldsymbol{x}, \mathrm{t})$, given measured covariates $\boldsymbol{x}$, observed treatment t, and model parameters $\boldsymbol{\omega}$. For continuous outcomes, a Gaussian likelihood can be used: $\mathcal{N}(y \mid \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}), \widehat{\sigma}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}))$. For discrete outcomes, a Bernoulli likelihood can be

used: $\text{Bern}(y \mid \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \text{t}))$. In both cases, $\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \text{t})$ is a parametric estimator of $\mathbb{E}[Y \mid \text{T} = \text{t}, \boldsymbol{X} = \boldsymbol{x}]$, which leads to: $\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x}) \equiv \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 1) - \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 0)$, a parametric CATE estimator.

Bayesian inference over the model parameters $\boldsymbol{\omega}$ treated as stochastic instances of the random variable $\boldsymbol{\Omega} \in \mathcal{W}$ has been shown by Jesson et al. [2020] to yield models capable of quantifying when assumption A.3 (overlap) does not hold, or when there is insufficient knowledge about the treatment effect $\tau(\boldsymbol{x})$ because the observed value $\boldsymbol{x}$ lies far from the support of $P_{\mathcal{D}}(\boldsymbol{X}, \text{T}, Y^0, Y^1)$. Such methods seek to enable sampling from the posterior distribution of the model parameters given the data, $p(\boldsymbol{\Omega} \mid \mathcal{D})$. Each sample, $\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D})$ induces a unique CATE function $\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x})$. Epistemic uncertainty is a measure of "disagreement" between the functions at a given value $\boldsymbol{x}$ [Kendall and Gal, 2017]. Jesson et al. [2020] propose $\text{Var}_{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D})}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 1) - \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 0))$ as a measure of epistemic uncertainty in the CATE.

A.1.2 Active Learning

We use a slightly different setup here than in the other chapters. Formally, an active learning setup consists of an unlabeled dataset $\mathcal{D}^{\text{pool}} = \{\boldsymbol{x}_i\}_{i=1}^{n_{\text{pool}}}$, a labeled training set $\mathcal{D}^{\text{train}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n_{\text{train}}}$, and a predictive model with likelihood $p_{\boldsymbol{\omega}}(y \mid \boldsymbol{x})$ parameterized by $\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})$. It is further assumed that an oracle exists to provide outcomes y for any data point in $\mathcal{D}^{\text{pool}}$. After model training, a batch of data $\{\boldsymbol{x}_i^*\}_{i=1}^{b}$ is selected from $\mathcal{D}^{\text{pool}}$ using an acquisition function a according to the informativeness of the batch.

We depart from the standard active learning setting and include the treatment: for active learning of treatment effects, we define $\mathcal{D}^{\text{pool}} = \{\boldsymbol{x}_i, \text{t}_i\}_{i=1}^{n_{\text{pool}}}$, a labeled training set $\mathcal{D}^{\text{train}} = \{\boldsymbol{x}_i, \text{t}_i, y_i\}_{i=1}^{n_{\text{train}}}$, and a predictive model with likelihood $p_{\boldsymbol{\omega}}(y \mid \boldsymbol{x}, \text{t})$ parameterized by $\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})$. The acquisition function takes as input $\mathcal{D}^{\text{pool}}$ and returns a batch of data $\{\boldsymbol{x}_i, \text{t}_i\}_{i=1}^{b}$ which are labeled using an oracle and added to $\mathcal{D}^{\text{train}}$. We focus on examining when there is only access to the observed treatments $\{\text{t}_i\}_{i=1}^{n_{\text{pool}}}$: scenarios where treatment assignment is not possible.

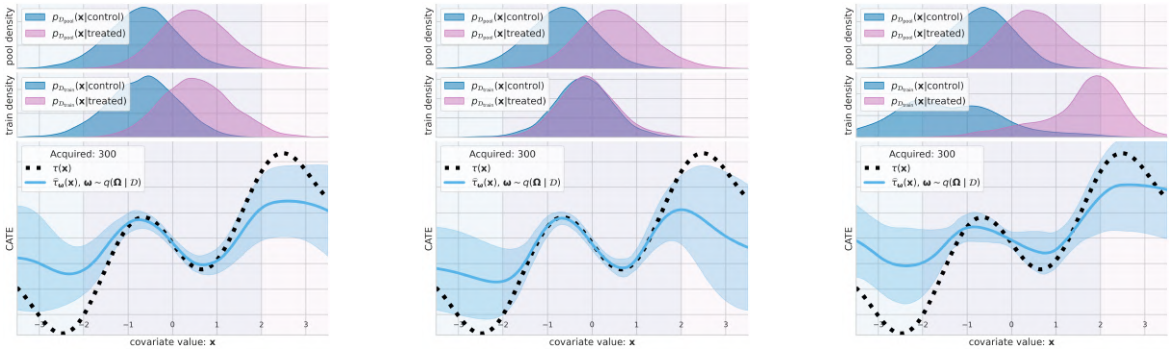
An intuitive way to define informativeness is using the estimated uncertainty of our model. In general, we can distinguish two sources of uncertainty: epistemic and aleatoric uncertainty [Der Kiureghian and Ditlevsen, 2009; Kendall and Gal, 2017]. Epistemic (or model) uncertainty, arises from uncertainty in the model parameters. This is for example caused by the model not having seen similar data points before, and therefore is unclear what the correct label would be. As before, we focus on using epistemic uncertainty to identify informative points to acquire the label for.

A.2 Methods

In this section, we introduce several acquisition functions, analyze how they bias the acquisition of training data, and show the resulting CATE functions learned from such training data. We are interested in acquisition functions conditioned on realizations of both $\boldsymbol{x}$ and t:

$$a(\mathcal{D}^{\text{pool}}, p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})) = \underset{\{\boldsymbol{x}_i, \text{t}_i\}_{i=1}^{b} \subseteq \mathcal{D}^{\text{pool}}}{\arg\max} \text{I}(\bullet \mid \{\boldsymbol{x}_i, \text{t}_i\}, \mathcal{D}^{\text{train}}),$$

where $\text{I}(\bullet \mid \boldsymbol{x}, \text{t}, \mathcal{D}^{\text{train}})$ is a measure of disagreement between parametric function predictions given $\boldsymbol{x}$ and t over samples $\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D})$. We make assumptions A.1 and A.2 (consistency, and unconfoundedness). We relax assumption A.3 (overlap) by



(a) Random **(b)** Propensity **(c)** τBALD

Figure A.2: Naive acquisition functions: How the training set is biased and how this effects the CATE function with a fixed budget of 300 acquired points.

allowing for its violation over subsets of the support of $\mathcal{D}^{\text{pool}}$. We present all theorems, proofs, and detailed assumptions in Appendix A.6.1.

A.2.1 Naive Acquisition Functions, Training Data Bias, and the Effect on the Estimated CATE Function.

To motivate Causal–BALD, we first look at a set of naive acquisition functions. The simplest function selects data points uniformly at random from $\mathcal{D}^{\text{pool}}$ and adds them to $\mathcal{D}^{\text{train}}$. In Figure A.2(a) we have acquired 300 such examples from a synthetic dataset and trained a deep-kernel Gaussian process [van Amersfoort et al., 2021] on those labeled examples. Comparing the top two panes, we see that $\mathcal{D}^{\text{train}}$ (middle) contains an unbiased sample of the data in $\mathcal{D}^{\text{pool}}$ (top). However, in the bottom pane we see that while the CATE estimator is accurate (and certain) near the modes of $\mathcal{D}^{\text{pool}}$, it becomes less accurate as we move to lower density regions. In this way random acquisition reflects the biases inherent in $\mathcal{D}^{\text{pool}}$ and over-allocates resources to the modes of the distribution. If the mode were to coincide with a region of non-overlap, the function would most frequently acquire uninformative examples.

Next, we look at using propensity scores to bias acquisition toward regions where the overlap assumption is satisfied.

Definition A.1. Propensity based acquisition

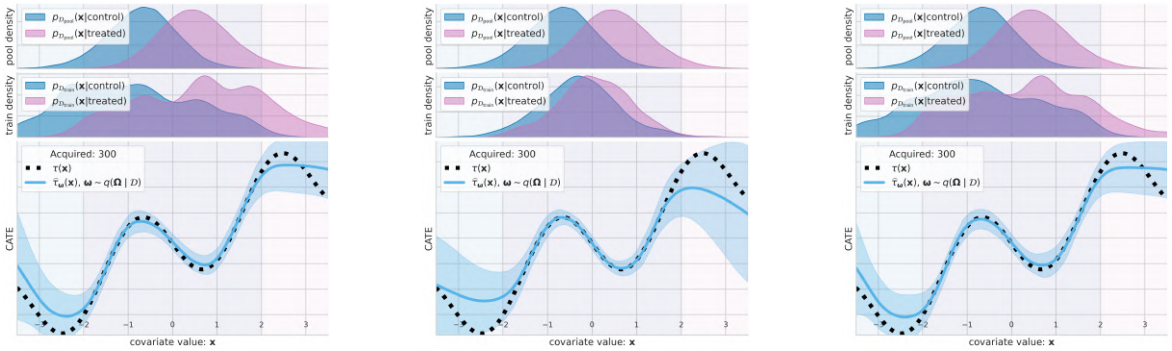
$$\mathrm{I}(\widehat{\pi}_{\mathrm{t}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) \equiv 1 - \widehat{\pi}_{\mathrm{t}}(\boldsymbol{x}) \tag{A.1}$$

Intuitively, this function prefers points where the propensity for observing the counterfactual is high. We are considering the setup where $\mathcal{D}^{\text{pool}}$ contains observations of both $\boldsymbol{X}$ and T, so it is straightforward to train an estimator for the propensity, $\widehat{\pi}_{\mathrm{t}}(\boldsymbol{x})$. Figure A.2(b) shows that while propensity score acquisition matches the treated and control densities in $\mathcal{D}^{\text{train}}$, it still biases acquisition towards the modes of $\mathcal{D}^{\text{pool}}$.

BALD aims to acquire data $(\boldsymbol{x}, \mathrm{t})$ that maximally reduce uncertainty in the model parameters $\boldsymbol{\Omega}$ used to predict the treatment effect. The most direct way to apply BALD is to use our uncertainty over the predicted treatment effect, expressed using the following information theoretic quantity:

Definition A.2. τBALD

$$\mathrm{I}(Y^1 - Y^0; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) \approx \operatorname*{Var}_{\boldsymbol{\omega}} \left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 1) - \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 0) \right)$$



(a) μBALD **(b)** ρBALD **(c)** $\mu\rho$BALD

Figure A.3: Causal–BALD acquisition functions: How the training set is biased and how this effects the CATE function with a fixed budget of 300 acquired points.

Building off the result in [Jesson et al., 2020], we show how the LHS measure about the *unobservable potential outcomes* can be estimated by the variance over $\mathbf{\Omega}$ of the *identifiable difference in expected outcomes* in Theorem A.1 of the appendix. A similar result has been proposed for non-parametric models Alaa and van der Schaar [2018]. Intuitively, this measure represents the information gain for $\mathbf{\Omega}$ if we could observe the difference in potential outcomes $Y^1 - Y^0$ for a given measurement $\boldsymbol{x}$ and $\mathcal{D}^{\text{train}}$.

However, labels for the random variable $Y^1 - Y^0$ are never observed so τBALD represents an irreducible measure of uncertainty. That is, τBALD will be high if it is uncertain about the label given the unobserved treatment t', regardless of its certainty about the label given the observed treatment t, which makes τBALD highest for low-density regions and regions with no overlap. Figure A.2(c) illustrates these consequences. We see the acquisition biases the training data away from the modes of the $\mathcal{D}^{\text{pool}}$, where we cannot know the treatment effect (no overlap). In datasets where we do not have full overlap, it leads to uninformative acquisitions.

A.2.2 Causal–BALD

In the previous section we looked at naive methods that either considered overlap, or considered information gain. In this section we develop a measure that take into account both factors when choosing a new training data point.

First, we focus only on reducible uncertainty:

Definition A.3. μBALD

$$\mathrm{I}(Y^{\mathrm{t}}; \mathbf{\Omega} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) \approx \underset{\boldsymbol{\omega} \sim p(\mathbf{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathrm{Var}} \left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}) \right). \tag{A.2}$$

This measure represents the information gain for the model parameters $\mathbf{\Omega}$ if we obtain a label for the observed potential outcome Y^{t} given a data point $(\boldsymbol{x}, \mathrm{t})$ and $\mathcal{D}^{\text{train}}$. Proof is given in Theorem A.4 of the appendix.

μBALD only contains observable quantities; however, it does not take into account our belief about the counterfactual outcome. As illustrated in Figure A.3(a), this approach can prefer acquiring $(\boldsymbol{x}, \mathrm{t})$ when we are also very uncertain about $(\boldsymbol{x}, \mathrm{t}')$, even if $(\boldsymbol{x}, \mathrm{t}')$ is not in $\mathcal{D}^{\text{pool}}$. Since we can neither reduce uncertainty over such $(\boldsymbol{x}, \mathrm{t}')$ nor know the treatment effect, acquisition would not be data efficient.

Next, we can take an information theoretic approach to combining knowledge about a data point's information gain and overlap. Let $\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t})$ be an instance of the random variable $\widehat{\mu}^{\mathrm{t}}_{\boldsymbol{\Omega}} \in \mathbb{R}$ corresponding to the expected outcome conditioned on t. Further, let $\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x})$ be an instance of the random variable $\widehat{\tau}_{\boldsymbol{\Omega}} = \widehat{\mu}^{1}_{\boldsymbol{\Omega}} - \widehat{\mu}^{0}_{\boldsymbol{\Omega}}$ corresponding to the CATE. Then,

Definition A.4. ρBALD

$$\mathrm{I}(Y^{\mathrm{t}}; \widehat{\tau}_{\boldsymbol{\Omega}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\mathrm{train}}) \gtrapprox \frac{1}{2} \log \left(\frac{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x}))}{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))} \right) \tag{A.3}$$

This measure represents the information gain for the CATE $\tau_{\boldsymbol{\Omega}}$ if we observe the outcome Y for a data point ($\boldsymbol{x}$, t) and the data we have trained on $\mathcal{D}^{\mathrm{train}}$. Proof for this result is given in Theorem A.5.

In contrast to μ-BALD, this measure accounts for overlap in two ways. First, ρ–BALD will be scaled by the inverse of the variance of the expected counterfactual outcome $\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}')$. This will bias acquisition towards examples for which we are certain about counterfactual outcome, and so we can assume that overlap is satisfied for observed $(\boldsymbol{x}, \mathrm{t})$. Second, ρ–BALD is discounted by $\mathrm{Cov}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}), \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))$. This is an interesting concept that we will leave for future discussion.

In Figure A.3(b) we see that ρ–BALD has matched the distributions of the treated and control groups in a similar manner to propensity acquisition in Figure A.2(b). Further, we see that the CATE estimator is more accurate over the support of the data. However, there is a shortcoming of ρ–BALD that results in it under exploring low density regions of $\mathcal{D}^{\mathrm{pool}}$, which we comment on in §A.6.1.3.

To combine the positive attributes of μ–BALD and ρ–BALD, while mitigating their shortcomings, we introduce $\mu\rho$BALD.

Definition A.5. $\mu\rho$BALD

$$\mathrm{I}(\mu\rho \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\mathrm{train}}) \equiv \underset{\boldsymbol{\omega}}{\mathrm{Var}} \left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}) \right) \frac{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x}))}{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))}.$$

Here, we scale Equation A.3, which has equivalent expression $\frac{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x}))}{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))}$ by our measure for μBALD such that in the cases where the ratio may be equal, there is a preference for data points the current model is more uncertain about. We can see in Figure A.3(c) that the acquisition of training data examples is more uniformly distributed over the support of the pool data where overlap is satisfied. Furthermore, the accuracy of the CATE estimator is the highest over that region.

A.3 Related Work

Deng et al. [2011] propose the use of Active Learning for recruiting patients to assign treatments that will reduce the uncertainty of an Individual Treatment Effect model. However, their setting is different from ours—we assume that suggesting treatments are too risky or even potentially lethal—and instead we acquire patients for the purpose of revealing their outcome (e.g. by having a biopsy). Additionally, although their method uses the predictive uncertainty to identify which patients to recruit, it does not disentangle the sources of uncertainty and as such treatments with high

outcome variance will be recruited as well. Closer to our proposal is the work from Sundin et al. [2019], where the authors propose the use of a Gaussian process (GP) to model the individual treatment effect and use the expected information gain over the S-type error rate, defined as the error in predicting the sign of the CATE, as their acquisition function. We compare to this in our experiment by limiting the access to counterfactual observations ($\boldsymbol{\gamma}$ baseline) and adapting it to Deep Ensembles [Lakshminarayanan et al., 2017] and DUE [van Amersfoort et al., 2021] (more details about the adaptation is provided in Appendix A.6.2.1).

A.4 Empirical Validation

In this section we evaluate our acquisition objectives on synthetic and semi-synthetic datasets.

Models Our objectives rely on methods that are capable of modeling the uncertainty and handling high-dimensional data modalities. DUE [van Amersfoort et al., 2021] is an instance of Deep Kernel Learning [Wilson et al., 2016], where a deep feature extractor is used to transform the inputs over which a Gaussian process' (GP) kernel is defined. In particular, DUE uses a variational inducing point approximation [Hensman et al., 2015] and a constrained feature extractor which contains residual connections and spectral normalisation to enable reliable uncertainty. It was previously shown to obtain SotA results on IHDP [van Amersfoort et al., 2021]. In DUE, we distinguish between the model parameters θ and the variational parameters $\boldsymbol{\omega}$, and we are Bayesian only over the $\boldsymbol{\omega}$ parameters. Since DUE is a GP, we obtain a full Gaussian posterior over our outputs from which we can use the mean and covariance directly. When necessary, sampling is very efficient and only requires a single forward pass in the deep model. We describe all hyper parameters in Appendix A.6.7.

Baselines We compare against the following baselines:

Random. This acquisition function selects points uniformly at random.

Propensity. An acquisition function based on the propensity score (Eq. A.1). We train a propensity model on the combination of the train and pool dataset which we then use acquire points based on their propensity score. Please note that this is a valid assumption as training a propensity model does not require outcomes.

$\boldsymbol{\gamma}$ (S-type error rate) [Sundin et al., 2019]. This acquisition function is the S-type error rate based method proposed by Sundin et al. [2019]. We have adapted the acquisition function to use with Bayesian Deep Neural Networks. The objective is defined as $\mathrm{I}(\boldsymbol{\gamma}; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}})$, where $\gamma(\boldsymbol{x}) = \text{probit}^{-1}(-\frac{|\mathbf{E}_{p(\tau|\boldsymbol{x},\mathcal{D}^{\text{train}})}[\tau]|}{\sqrt{\text{Var}(\tau|\boldsymbol{x},\mathcal{D}^{\text{train}})}})$ and $\text{probit}^{-1}(\cdot)$ is the cumulative distribution function of normal distribution. In contrast to the original formulation, we do not assume access to counterfactual observations at training time.

Datasets Starting from the hypothesis that different objectives can target different types of imbalances and overlap ratios we construct a **synthetic** dataset [Kallus et al., 2019] demonstrating the different biases. Additionally, we study the performance of our acquisition functions on the **IHDP** dataset [Hill, 2011; Shalit et al., 2017], a standard benchmark in causal treatment effect literature, and finally we demonstrate that our

method is suitable for high dimensional datasets on **CMNIST** [Jesson et al., 2021], an MNIST [LeCun, 1998] based dataset adapted for causal treatment effect studies. Detailed descriptions of each dataset are given in Appendix A.6.3.

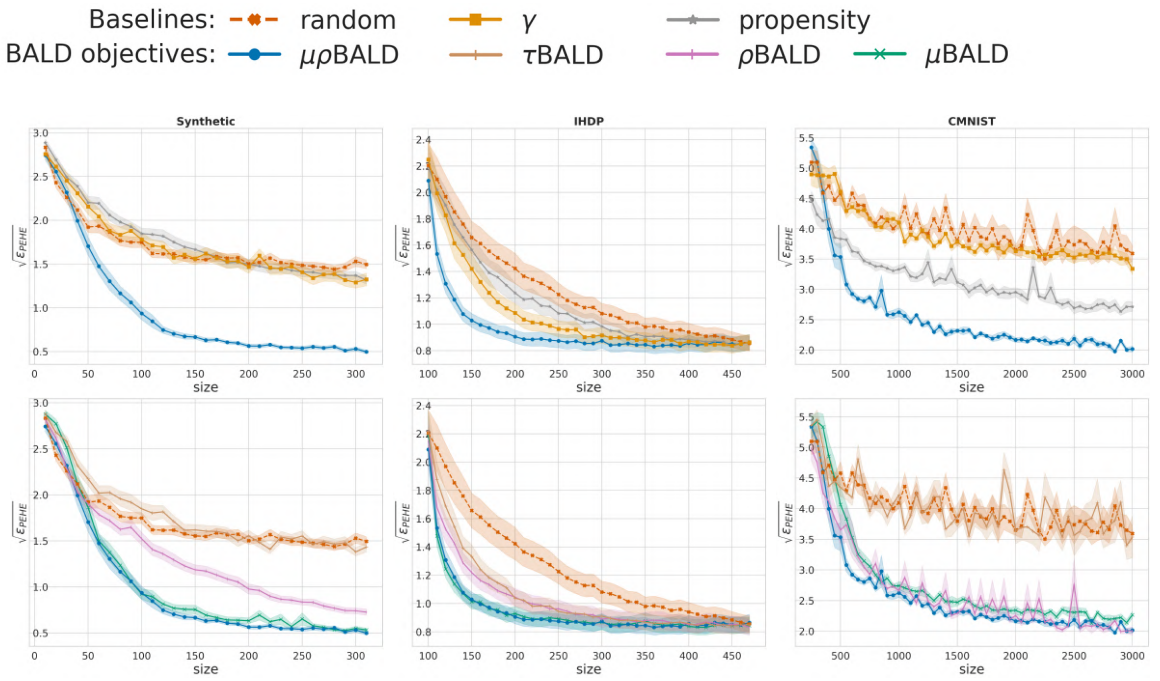


Figure A.4: $\sqrt{\epsilon_{PEHE}}$ performance (shaded standard error) for DUE models. **(left to right) synthetic** (40 seeds), and **IHDP** (200 seeds). We observe that BALD objectives outperform the **random**, $\boldsymbol{\gamma}$ and **propensity** acquisition functions significantly, suggesting that epistemic uncertainty aware methods that target reducible uncertainty can be more sample efficient.

A.4.1 Results

For each of the acquisition objectives, dataset, and model we present the mean and standard error of empirical square root of precision in estimation of heterogenous effect (PEHE) [1]. We summarize each active learning setup in Table A.1.

In Figure A.4, we see that epistemic uncertainty aware $\mu\rho$BALD outperforms the baselines, random, propensity and S-Type error rate ($\boldsymbol{\gamma}$). As we analysed in section A.2, this is expected as our acquisition objectives target the type of uncertainty that can be reduced – that is the epistemic uncertainty for which we have overlap between treatment and control. Additionally, $\mu\rho$BALD shows superior performance over the other objectives in the high dimensional dataset CMNIST verifying our qualitative analysis in Figure A.3(c).

A.5 Discussion

We have introduced a new acquisition function for active learning of individual-level causal-treatment effects from high dimensional observational data, based on Bayesian Active Learning by Disagreement Houlsby et al. [2011]. We derive our proposed method from an information theoretic perspective and compared with various acquisition

[1] $\sqrt{\epsilon_{PEHE}} = \sqrt{\frac{1}{N}\sum_{\boldsymbol{x}}(\hat{\tau}(\boldsymbol{x}) - \tau(\boldsymbol{x}))^2}$

functions that do not take into consideration epistemic uncertainty (like random or propensity based) or they target uncertainties that cannot be reduced in the observational setting (i.e. when we do not have access to counterfactual observations). We show that our methods significantly outperform the baselines while also studying the various properties of each of our proposed objectives in both a quantitative and a qualitative analysis, potentially impacting areas like healthcare where sample efficiency in acquisition of new examples imply improved safety and reductions in costs.

A.6 Details

A.6.1 Theoretical Results

A.6.1.1 τ-BALD

Theorem A.1. *Under the following assumptions:*

1. *Unconfoundedness* $(Y^0, Y^1) \perp\!\!\!\perp \mathrm{T} \mid \boldsymbol{X}$;
2. *Consistency* $Y \mid \mathrm{T} = Y^{\mathrm{t}}$;
3. Y^1 *and* Y^0*, when conditioned on realizations* $\boldsymbol{x}$ *of the r.v.* $\boldsymbol{X}$ *and* t *of the r.v.* T*, are independent-normally distributed or joint-normally distributed r.v.s.*
4. $\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t})$ *is a consistent estimator of* $\mathbb{E}[Y \mid \mathrm{T} = \mathrm{t}, \boldsymbol{X} = \boldsymbol{x}]$

the information gain for $\boldsymbol{\Omega}$ *if we could observe a label for the difference in potential outcomes* $Y^1 - Y^0$ *given measured covariates* $\boldsymbol{x}$*, treatment* t *and a dataset of observations* $\mathcal{D}^{\text{train}} = \{\boldsymbol{x}_i, \mathrm{t}_i, y_i\}_{i=1}^n$ *is approximated as*

$$\mathrm{I}(Y^1 - Y^0; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) \approx \underset{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathrm{Var}} \left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 1) - \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 0)\right) \tag{A.4}$$

Proof.

$$\mathrm{I}(Y^1 - Y^0; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}) = \mathrm{H}(Y^1 - Y^0 \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}) - \underset{p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathbb{E}} \Big[\mathrm{H}(Y^1 - Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega})\Big] \tag{A.5a}$$

$$\approx \mathrm{Var}(Y^1 - Y^0 \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}) - \underset{p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathbb{E}} \Big[\mathrm{Var}(Y^1 - Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega})\Big] \tag{A.5b}$$

$$= \underset{p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathrm{Var}} \Big(\mathbb{E}[Y^1 - Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega}]\Big) \tag{A.5c}$$

$$= \underset{p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathrm{Var}} \left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 1) - \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 0)\right) \tag{A.5d}$$

In (A.5a) we adapt the result of Houlsby et al. [2011] and express the information gain as the mutual information between the observable difference in potential outcomes $Y^1 - Y^0$ and the parameters $\boldsymbol{\Omega}$; given observed covariates $\boldsymbol{x}$, treatment t, and training data $\mathcal{D}^{\text{train}} = \{(\boldsymbol{x}_i, \mathrm{t}_i, y_i)\}_{i=1}^{n_{\text{train}}}$. In (A.5b) we apply lemma A.2 to the r.h.s terms of (A.5a). We then use the result in Jesson et al. [2020] and move from (A.5b) to (A.5c) by application of the law of total variance. Finally, under the consistency and unconfoundedness assumptions we express the information gain in terms of the identifiable difference in expected outcomes $\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 1) - \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 0)$. □

Lemma A.2. *Under the following assumptions:*

1. Y^1, Y^0 *are independent-normally distributed or joint-normally distributed r.v.s;*

2. *With* $A = \mathrm{Var}(Y^1 - Y^0)$*: let* $|A - 1| \leq 1$ *and* $A \neq 0$*. That is to say, the predictive variance must be greater than 0 and less than or equal to 2;*

$$\mathrm{H}(Y^1 - Y^0) \approx \mathrm{Var}(Y^1 - Y^0) \tag{A.6}$$

Proof. By assumption 1, $Y^1 - Y^0$ is also a normally distributed random variable. By corollary A.3,

$$\mathrm{H}(Y^1 - Y^0) = \frac{1}{2} + \frac{1}{2}\log(2\pi\,\mathrm{Var}(Y^1 - Y^0)) \tag{A.7}$$

So given assumption 2, the first order Taylor polynomial of $\mathrm{H}(Y^1 - Y^0)$ is

$$\begin{aligned} \frac{1}{2} + \frac{1}{2}\log(2\pi\,\mathrm{Var}(Y^1 - Y^0)) &\approx \frac{1}{2} + \frac{1}{2}(2\pi\,\mathrm{Var}(Y^1 - Y^0) - 1) \\ &= \frac{1}{2} + \pi\,\mathrm{Var}(Y^1 - Y^0) - \frac{1}{2} \\ &= \pi\,\mathrm{Var}(Y^1 - Y^0) \\ &\propto \mathrm{Var}(Y^1 - Y^0) \end{aligned} \tag{A.8}$$

□

Corollary A.3. *The entropy of a normally distributed random variable with variance* σ^2 *is* $\frac{1}{2} + \frac{1}{2}\log(2\pi\sigma^2)$

A.6.1.2 μ-BALD

Theorem A.4. *Under the following assumptions:*

1. *Unconfoundedness* $(Y^0, Y^1) \perp\!\!\!\perp \mathrm{T} \mid \boldsymbol{X}$*,*
2. *Consistency* $Y \mid \mathrm{T} = Y^{\mathrm{t}}$*,*
3. Y *conditioned on* $\boldsymbol{x}$ *and* t *is a normally distributed random variable,*
4. $\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t})$ *is a consistent estimator of* $\mathbb{E}[Y \mid \mathrm{T} = \mathrm{t}, \boldsymbol{X} = \boldsymbol{x}]$*,*

the information gain for $\boldsymbol{\Omega}$ *when we observe a label for the potential outcome* Y^{t} *given measured covariates* $\boldsymbol{x}$*, treatment* t *and a dataset of observations* $\mathcal{D}^{\text{train}} = \{\boldsymbol{x}_i, \mathrm{t}_i, y_i\}_{i=1}^n$ *can be approximated as is*

$$\mathrm{I}(Y^{\mathrm{t}}; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) \approx \frac{1}{2}\log\left(\frac{\mathrm{Var}(Y \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}})}{\mathbb{E}_{\boldsymbol{\omega}}[\mathrm{Var}(Y \mid \boldsymbol{x}, \mathrm{t}, \boldsymbol{\omega})]}\right), \tag{A.9}$$

or

$$\mathrm{I}(Y^{\mathrm{t}}; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) \approx \underset{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathrm{Var}} \left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t})\right). \tag{A.10}$$

Equation (A.9) expresses the information gain as the logarithm of a ratio between predictive and aleatoric uncertainty in the outcome. Whereas, equation (A.9) expresses the information gain as a direct estimate of the epistemic uncertainty.

Proof.

$$\mathrm{I}(Y^{\mathrm{t}}; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) = \mathrm{H}(Y \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) - \underset{p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathbb{E}} [\mathrm{H}(Y \mid \boldsymbol{x}, \mathrm{t}, \boldsymbol{\omega})] \tag{A.11a}$$

$$= \frac{1}{2}\log\left(2\pi\,\mathrm{Var}(Y \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}})\right) - \underset{p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathbb{E}} \frac{1}{2}\log\left(2\pi\,\mathrm{Var}(Y \mid \boldsymbol{\omega}, \boldsymbol{x}, \mathrm{t})\right) \tag{A.11b}$$

$$\geq \frac{1}{2}\log\left(2\pi \operatorname{Var}(Y \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}})\right) - \frac{1}{2}\log\left(2\pi \mathop{\mathbb{E}}_{p(\boldsymbol{\Omega}|\mathcal{D}^{\text{train}})} \operatorname{Var}(Y \mid \boldsymbol{\omega}, \boldsymbol{x}, \mathrm{t})\right) \tag{A.11c}$$

$$= \frac{1}{2}\log\left(\frac{\operatorname{Var}(Y \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}})}{\mathbb{E}_{\boldsymbol{\omega}}[\operatorname{Var}(Y \mid \boldsymbol{x}, \mathrm{t}, \boldsymbol{\omega})]}\right) \tag{A.11d}$$

$$\mathrm{I}(Y^{\mathrm{t}}; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) = \mathrm{H}(Y \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) - \mathop{\mathbb{E}}_{p(\boldsymbol{\Omega}|\mathcal{D}^{\text{train}})}[\mathrm{H}(Y \mid \boldsymbol{x}, \mathrm{t}, \boldsymbol{\omega})] \tag{A.12a}$$

$$\approx \operatorname{Var}[Y \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\Omega}|\mathcal{D}^{\text{train}})}[\operatorname{Var}[Y \mid \boldsymbol{x}, \mathrm{t}, \boldsymbol{\omega}]] \tag{A.12b}$$

$$= \mathop{\operatorname{Var}}_{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega}|\mathcal{D}^{\text{train}})}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t})) \tag{A.12c}$$

In (A.12a) we express the information gain as the mutual information between the observed potential outcome Y^{t} and the parameters $\boldsymbol{\Omega}$; given observed covariates $\boldsymbol{x}$, treatment t, and training data $\mathcal{D}^{\text{train}}$. By consistency, we can drop the superscript on the potential outcome. In (A.12b) we approximate the r.h.s terms of (A.12a) by application of Lemma A.2. Finally, we can move from (A.12b) to (A.12c) by application of the law of total variance. □

Note that for discrete or categorical Y, it is straightforward to evaluate Equation (A.12a) directly.

A.6.1.3 ρ-BALD

Theorem A.5. *Under the following assumptions*

1. $\{\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}) : \mathrm{t} \in \{0, 1\}\}$ *are instances of the independent-normally distributed or joint-normally distributed random variables* $\{\widehat{\mu}^{\mathrm{t}}_{\boldsymbol{\Omega}} = \mathbb{E}[Y \mid \boldsymbol{\Omega}, \mathrm{T} = \mathrm{t}, \boldsymbol{x}] : \mathrm{t} \in \{0, 1\}\}$,
2. $\operatorname{Var}_{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega}|\mathcal{D}^{\text{train}})}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}')) > 0$.

Let $\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x})$ be a realization of the random variable $\widehat{\tau}_{\boldsymbol{\Omega}} = \widehat{\mu}^1_{\boldsymbol{\Omega}} - \widehat{\mu}^0_{\boldsymbol{\Omega}}$. The information gain for $\widehat{\tau}_{\boldsymbol{\Omega}}$ if we observe the label for the potential outcome Y^{t} given measured covariates $\boldsymbol{x}$, treatment t *and a dataset of observations $\mathcal{D}^{\text{train}} = \{\boldsymbol{x}_i, \mathrm{t}_i, y_i\}_{i=1}^n$ is approximately*

$$\begin{aligned} \mathrm{I}(Y^{\mathrm{t}}; \widehat{\tau}_{\boldsymbol{\Omega}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) &\approx \frac{\operatorname{Var}_{\boldsymbol{\omega}}(\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x}))}{\operatorname{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))}, \\ &= \frac{\operatorname{Var}_{\boldsymbol{\omega}}\left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t})\right) - 2\operatorname{Cov}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}), \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))}{\operatorname{Var}_{\boldsymbol{\omega}}\left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}')\right)} + 1, \end{aligned} \tag{A.13}$$

where for binary $\mathrm{T} = t$, $\mathrm{t}' = (1 - \mathrm{t})$.

Proof.

$$\mathrm{I}(Y^{\mathrm{t}}; \widehat{\tau}_{\boldsymbol{\Omega}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}) = \mathrm{H}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}) - \mathrm{H}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid Y^{\mathrm{t}}, \boldsymbol{x}, \mathrm{t}, \mathcal{D}) \tag{A.14a}$$

$$= \mathrm{H}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}) - \mathop{\mathbb{E}}_{y^{\mathrm{t}} \sim p(Y^{\mathrm{t}}|\boldsymbol{x}, \mathrm{t}, \mathcal{D})} \mathrm{H}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid y^{\mathrm{t}}, \boldsymbol{x}, \mathrm{t}) \tag{A.14b}$$

$$= \frac{1}{2}\log(2\pi \operatorname{Var}(\widehat{\tau}_{\boldsymbol{\Omega}})) - \mathop{\mathbb{E}}_{y^{\mathrm{t}} \sim p(Y^{\mathrm{t}}|\boldsymbol{x}, \mathrm{t}, \mathcal{D})}\left[\frac{1}{2}\log(2\pi \operatorname{Var}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid y^{\mathrm{t}}))\right] \tag{A.14c}$$

$$\geq \frac{1}{2}\log(2\pi \operatorname{Var}(\widehat{\tau}_{\boldsymbol{\Omega}})) - \frac{1}{2}\log(2\pi\, \mathbb{E}\left[\operatorname{Var}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid y^{\mathrm{t}})\right]) \tag{A.14d}$$

$$= \frac{1}{2} \log \left(\frac{\mathrm{Var}(\widehat{\tau}_{\boldsymbol{\Omega}})}{\mathbb{E}\left[\mathrm{Var}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid y^{\mathrm{t}})\right]} \right), \tag{A.14e}$$

and we can further expand the fraction to

$$\frac{\mathrm{Var}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D})}{\mathbb{E}\left[\mathrm{Var}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid y^{\mathrm{t}})\right]} = \tag{A.14f}$$

$$= \frac{\mathrm{Var}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D})}{\mathrm{Var}_{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D})}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))} \tag{A.14g}$$

$$= \frac{\mathrm{Var}_{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D})}(\widehat{\tau}_{\boldsymbol{\omega}}(\boldsymbol{x}) \mid \mathrm{t})}{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))} \tag{A.14h}$$

$$= \frac{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 1) - \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, 0) \mid \mathrm{t})}{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))} \tag{A.14i}$$

$$= \frac{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}) - \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))}{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))} \tag{A.14j}$$

$$= \frac{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t})) + \mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}')) - 2\,\mathrm{Cov}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}), \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))}{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))} \tag{A.14k}$$

$$= \frac{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t})) - 2\,\mathrm{Cov}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}), \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))}{\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}'))} + 1, \tag{A.14l}$$

where (A.14a) by definition of mutual information; (A.14a)-(A.14b) from the result of Houlsby et al. [2011]; (A.14b)-(A.14c) by Assumption 1. and Corollary A.3; (A.14c)-(A.14d) by Jensen's inequality; (A.14d)-(A.14e) by the logarithmic quotient identity; (A.14f)-(A.14g) by Lemma A.6; (A.14g)-(A.14h) by definition of the variance. (A.14h)-(A.14i) by definition of $\widehat{\tau}_{\boldsymbol{\omega}}$; (A.14i)-(A.14j) by symmetry of the variance of the difference of two random variables; (A.14j)-(A.14k) by the definition of the variance of the difference of two random variables; and (A.14k)-(A.14l) by cancelling terms. □

Lemma A.6. *Under the following assumptions*

1. *Consistency* $Y \mid \mathrm{T} = Y^{\mathrm{t}}$;
2. *Unconfoundedness* $(Y^0, Y^1) \perp\!\!\!\perp \mathrm{T} \mid \boldsymbol{X}$;

$$\mathop{\mathbb{E}}_{y^{\mathrm{t}} \sim p(Y^{\mathrm{t}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D})} \left[\mathrm{Var}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid y^{\mathrm{t}}) \right] \approx \mathop{\mathbb{E}}_{y^{\mathrm{t}} \sim p(Y^{\mathrm{t}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D})} \left[\mathop{\mathrm{Var}}_{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D}^{\mathrm{train}})} (\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}')) \right], \tag{A.15}$$

where for binary $\mathrm{T} = \mathrm{t}$, $\mathrm{t}' = (1 - \mathrm{t})$.

Proof.

$$\mathbb{E}_{y^{\mathrm{t}} \sim p(Y^{\mathrm{t}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D})} \left[\mathrm{Var}(\widehat{\tau}_{\boldsymbol{\Omega}} \mid y^{\mathrm{t}}) \right] = \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\left(\widehat{\tau}_{\boldsymbol{\omega}} - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\widehat{\tau}_{\boldsymbol{\omega}} \mid y^{\mathrm{t}} \right] \right)^2 \mid y^{\mathrm{t}} \right] \right], \tag{A.16}$$

$$= \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\left(\mathbb{E}[Y^1 - Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathbb{E}[Y^1 - Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega}] \mid y^{\mathrm{t}} \right] \right)^2 \mid y^{\mathrm{t}} \right] \right], \tag{A.17}$$

$$= \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\left(\mathbb{E}[Y^1 \mid \boldsymbol{x}, \boldsymbol{\omega}] - \mathbb{E}[Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathbb{E}[Y^1 \mid \boldsymbol{x}, \boldsymbol{\omega}] \mid y^{\mathrm{t}} \right] + \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathbb{E}[Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega}] \mid y^{\mathrm{t}} \right] \right)^2 \mid y^{\mathrm{t}} \right] \right], \tag{A.18}$$

$$= \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\left(\left(\mathbb{E}[Y^1 \mid \boldsymbol{x}, \boldsymbol{\omega}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathbb{E}[Y^1 \mid \boldsymbol{x}, \boldsymbol{\omega}] \mid y^{\mathrm{t}} \right] \right) - \left(\mathbb{E}[Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathbb{E}[Y^0 \mid \boldsymbol{x}, \boldsymbol{\omega}] \mid y^{\mathrm{t}} \right] \right) \right)^2 \mid y^{\mathrm{t}} \right] \right], \tag{A.19}$$

$$= \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\left(\left(\mathbb{E}[Y^{\mathrm{t}} \mid \boldsymbol{x}, \boldsymbol{\omega}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathbb{E}[Y^{\mathrm{t}} \mid \boldsymbol{x}, \boldsymbol{\omega}] \mid y^{\mathrm{t}} \right] \right) - \left(\mathbb{E}[Y^{\mathrm{t}'} \mid \boldsymbol{x}, \boldsymbol{\omega}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathbb{E}[Y^{\mathrm{t}'} \mid \boldsymbol{x}, \boldsymbol{\omega}] \mid y^{\mathrm{t}} \right] \right) \right)^2 \mid y^{\mathrm{t}} \right] \right], \tag{A.20}$$

$$= \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega} \mid y^{\mathrm{t}})} \left[\left(\left(\mathop{\mathbb{E}}_{p(y^{\mathrm{t}} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega} \mid y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(y^{\mathrm{t}} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}}] \right] \right) - \left(\mathop{\mathbb{E}}_{p(y^{\mathrm{t}'} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}'}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega} \mid y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(y^{\mathrm{t}'} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}'}] \right] \right) \right)^2 \right] \right], \tag{A.21}$$

$$= \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega} \mid y^{\mathrm{t}})} \left[\left(\underbrace{\left(\mathop{\mathbb{E}}_{p(y^{\mathrm{t}} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega} \mid y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(y^{\mathrm{t}} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}}] \right] \right)}_{\approx 0} - \left(\mathop{\mathbb{E}}_{p(y^{\mathrm{t}'} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}'}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathop{\mathbb{E}}_{p(y^{\mathrm{t}'} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}'}] \right] \right) \right)^2 \right] \right], \tag{A.22}$$

$$\approx \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega} \mid y^{\mathrm{t}})} \left[\left(\mathop{\mathbb{E}}_{p(y^{\mathrm{t}'} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}'}] - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} \left[\mathop{\mathbb{E}}_{p(y^{\mathrm{t}'} \mid \boldsymbol{x}, \boldsymbol{\omega})} [y^{\mathrm{t}'}] \right] \right)^2 \right] \right], \tag{A.23}$$

$$= \mathop{\mathbb{E}}_{p(y^{\mathrm{t}})} \left[\mathop{\mathbb{E}}_{p(\boldsymbol{\omega} \mid y^{\mathrm{t}})} \left[\left(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}') - \mathop{\mathbb{E}}_{p(\boldsymbol{\omega})} [\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}')] \right)^2 \right] \right], \tag{A.24}$$

$$= \mathop{\mathbb{E}}_{y^{\mathrm{t}} \sim p(Y^{\mathrm{t}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D})} \left[\mathop{\mathrm{Var}}_{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D}^{\mathrm{train}})} (\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}')) \right], \tag{A.25}$$

where (A.16) by definition of variance; (A.16)-(A.17) by definition of $\widehat{\tau}_{\boldsymbol{\omega}}$; (A.17)-(A.18) by linearity of expectations; (A.18)-(A.19) by grouping terms; (A.19)-(A.20) by symmetry of the square; (A.20)-(A.21) by rewriting expectations in terms of densities; (A.21)-(A.22) the observed potential outcome does not have an effect on the expectation of the model for the counterfactual outcome; (A.22)-(A.23) we drop the term as an approximation as we cannot estimate here how much the expected outcome is going

to change—the conservative assumption is that will not change; (A.23)-(A.24) by definition of $\widehat{\mu}_{\boldsymbol{\omega}}$; (A.24)-(A.25) by definition of variance; □

ρ-BALD Failure Consider two examples in $\mathcal{D}^{\text{pool}}$, $(\boldsymbol{x}_1, \mathrm{t}_1)$ and $(\boldsymbol{x}_2, \mathrm{t}_2)$ where $\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_1, \mathrm{t}_1)) = \mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_1, \mathrm{t}_1'))$ and $\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_2, \mathrm{t}_2)) = \mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_2, \mathrm{t}_2'))$. That is, for each point we are as uncertain about the conditional expectation given the observed treatment as we would be given the counterfactual treatment. Further, let $\mathrm{Cov}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_1, \mathrm{t}_1), \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_1, \mathrm{t}_1')) = \mathrm{Cov}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_2, \mathrm{t}_2), \widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_2, \mathrm{t}_2'))$ and $\mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_1, \mathrm{t}_1)) > \mathrm{Var}_{\boldsymbol{\omega}}(\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}_2, \mathrm{t}_2))$. In this scenario ρ–BALD would rank these two points equally, but in practice it may be preferable to choose $(\boldsymbol{x}_1, \mathrm{t}_1)$ over $(\boldsymbol{x}_2, \mathrm{t}_2)$ as it would more likely be a point as yet unseen by the model. When naively acquiring multiple points per acquisition step, this method biases training data to the modes of $\mathcal{D}^{\text{pool}}$.

A.6.1.4 $\mu\pi$BALD

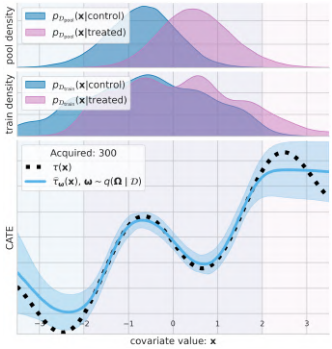


Figure A.5: $\mu\pi$BALD

The most straightforward way to combine knowledge about a data point's information gain and overlap is to simply multiply μBALD(A.2) by the propensity acquisition term (A.1):

Definition A.6. $\mu\pi$BALD

$$\mathrm{I}(\mu\pi \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}^{\text{train}}) \equiv (1 - \widehat{\pi}_{\mathrm{t}}(\boldsymbol{x})) \underset{\boldsymbol{\omega} \sim p(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})}{\mathrm{Var}} (\widehat{\mu}_{\boldsymbol{\omega}}(\boldsymbol{x}, \mathrm{t}))\,.$$

We can see in Figure A.5 that the acquisition of training data results in matched sampling as we saw for propensity acquisition in Figure A.2(b), but that the tails of the overlapping distributions extend further into the low density regions of the pool set support where overlap is satisfied.

A.6.2 Baselines

A.6.2.1 S-type Error Information Gain

In their work, Sundin et al. [2019] assume that the underlying model is a Gaussian Process (GP) and also that they have access to the counterfactual outcome. Although GPs are suitable for uncertainty estimation, they do not scale up to high dimensional datasets (e.g. images). We propose to use Deep Ensembles and DUE for alleviating the capabilities issues and we modified the objective to be more suitable for our architecture.

Following the formulation from Houlsby et al. [2011], the acquisition strategy becomes $\arg\max_{\boldsymbol{x}} \mathbb{H}[\gamma|\boldsymbol{x}, D] - \mathbb{E}_{\mathbb{H}[p(\theta|D)]}[\gamma|\boldsymbol{x}, \theta]]$, where $\gamma(\boldsymbol{x}) = \text{probit}^{-1}(-\frac{|\mathbf{E}_{p(\tau|\boldsymbol{x},\mathcal{D}^{\text{train}})}[\tau]|}{\sqrt{\text{Var}(\tau|\boldsymbol{x},\mathcal{D}^{\text{train}})}})$, $\text{probit}^{-1}(\cdot)$ is the cumulative distribution function of normal distribution and $p(\gamma|\boldsymbol{x}, D) = \text{Bernoulli}(\gamma)$. With DUE (Deep Kernel Learning method) Deep Ensembles (samples from $p(\theta|D)$ we can compute those terms similarly to how we implemented our BALD objectives.

Below is an example of how this was implemented in PyTorch:

```
tau_mu = mu1s - mu0s
tau_var = var1s + var0s + 1e-07
gammas = torch.distributions.normal.Normal(0, 1).cdf(
    -tau_mu.abs() / tau_var.sqrt()
)
gamma = gammas.mean(-1)
predictive_entropy = dist.Bernoulli(gamma).entropy()
conditional_entropy = dist.Bernoulli(gammas).entropy().mean(-1)
# it can get negative very small number
# because of numerical instabilities
scores = (predictive_entropy - conditional_entropy).clamp_min(1e-07)
```

A.6.3 Datasets

A.6.3.1 Synthetic Data

We modify the synthetic dataset presented by Kallus et al. [2019]. Our dataset is described by the following structural causal model (SCM):

$$\boldsymbol{x} \triangleq N_{\boldsymbol{x}}, \tag{A.26a}$$

$$\mathrm{t} \triangleq N_{\mathrm{t}}, \tag{A.26b}$$

$$y \triangleq (2\mathrm{t}-1)\boldsymbol{x} + (2\mathrm{t}-1) - 2\sin(2(2\mathrm{t}-1)\boldsymbol{x}) + 2(1+0.5\boldsymbol{x}) + N_y, \tag{A.26c}$$

where $N_{\boldsymbol{x}} \sim \mathcal{N}(0,1)$, $N_{\mathrm{t}} \sim \text{Bern}(\text{sigmoid}(2\boldsymbol{x}+0.5))$, and $N_y \sim \mathcal{N}(0,1)$.

Each random realization of the simulated dataset generates 10000 pool set examples, 1000 validation examples, and 1000 test examples. In the experiments we report results over 20 random realizations. The seeds for the random number generators are i, $i+1$, and $i+2$; $\{i \in [0, 1, \ldots, 19]\}$, for the training, validation, and test sets, respectively.

A.6.3.2 IHDP Data.

Infant Health and Development Program (IHDP) is a semisynthetic dataset [Hill, 2011; Shalit et al., 2017] commonly used in literature to study the performance of causal effect estimation methods. The dataset consists of 747 cases, out of which 139 are assigned in treatment group and 608 in control. Each unit is represented by 25 covariates describing different aspects of the infants and their mothers.

A.6.3.3 CMNIST Data.

Following the setup from Jesson et al. [2021], we use a simulated dataset based on MNIST [LeCun, 1998]. CMNIST is described by the following SCM:

$$\boldsymbol{x} \triangleq N_{\boldsymbol{x}}, \tag{A.27a}$$

$$\phi \triangleq \left(\text{clip}\left(\frac{\mu_{N_{\boldsymbol{x}}} - \mu_{\text{c}}}{\sigma_{\text{c}}}; -1.4, 1.4\right) - \text{Min}_{\text{c}}\right) \frac{\text{Max}_{\text{c}} - \text{Min}_{\text{c}}}{1.4 - \text{-}1.4} \tag{A.27b}$$

$$\text{t} \triangleq N_{\text{t}}, \tag{A.27c}$$

$$y \triangleq (2\text{t} - 1)\phi + (2\text{t} - 1) - 2\sin(2(2\text{t} - 1)\phi) + 2(1 + 0.5\phi) + N_y, \tag{A.27d}$$

where N_{t} (swapping x for ϕ), and N_y are as described in Appendix A.6.3.1. $N_{\boldsymbol{x}}$ is a sample of an MNIST image. The sampled image has a corresponding label $c \in [0, \ldots, 9]$. $\mu_{N_{\boldsymbol{x}}}$ is the average intensity of the sampled image. μ_{c} and σ_{c} are the mean and standard deviation of the average image intensities over all images with label c in the MNIST training set. In other words, $\mu_{\text{c}} = \mathbb{E}[\mu_{N_{\boldsymbol{x}}} \mid \text{c}]$ and $\sigma_{\text{c}}^2 = \text{Var}[\mu_{N_{\boldsymbol{x}}} \mid \text{c}]$. To map the high dimensional images $\boldsymbol{x}$ onto a one-dimensional manifold ϕ with domain $[-3, 3]$ above, we first clip the standardized average image intensity on the range $(-1.4, 1.4)$. Each digit class has its own domain in ϕ, so there is a linear transformation of the clipped value onto the range $[\text{Min}_{\text{c}}, \text{Max}_{\text{c}}]$. Finally, $\text{Min}_{\text{c}} = -2 + \frac{4}{10}\text{c}$, and $\text{Max}_{\text{c}} = -2 + \frac{4}{10}(\text{c} + 1)$.

For each random realization of the dataset, the MNIST training set is split into training ($n = 35000$) and validation ($n = 15000$) subsets using the scikit-learn function train_test_split(). The test set is generated using the MNIST test set ($n = 10000$). The random seeds are $\{i \in [0, 1, \ldots, 19]\}$ for the 20 random realizations generated.

A.6.4 Active Learning Setup Details

Table A.1: Summary of active learning setup per dataset.

Dataset	Warm up size	Acq. size	# of Acq.
Synthetic	0	10	31
IHDP	100	10	38 (max)
CMNIST	250	50	20

A.6.5 More Results

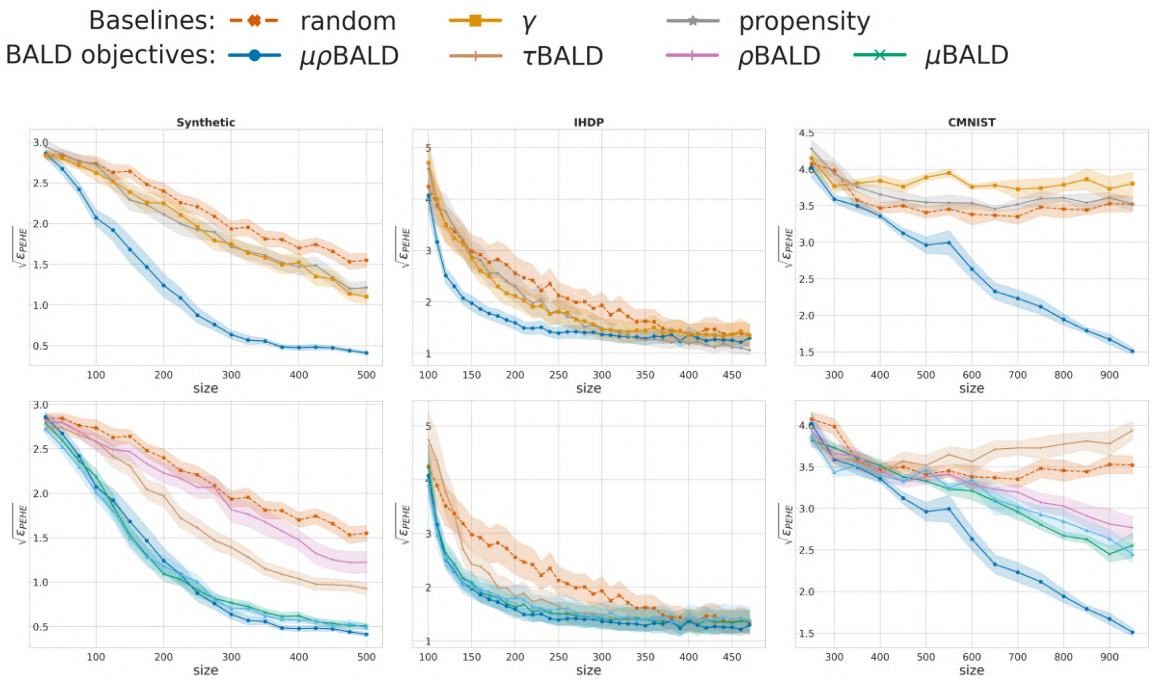


Figure A.6: $\sqrt{\epsilon_{PEHE}}$ performance (shaded standard error) for Deep Ensembles based models. **(left to right) synthetic** (20 seeds), **IHDP** (50 seeds) and *CMNIST* (5 seeds) dataset results, **(top to bottom)** comparison with baselines, comparison between BALD objectives. We observe that BALD objectives outperform the **random**, γ and **propensity** acquisition functions significantly, suggesting that epistemic uncertainty aware methods that target reducible uncertainty can be more sample efficient.

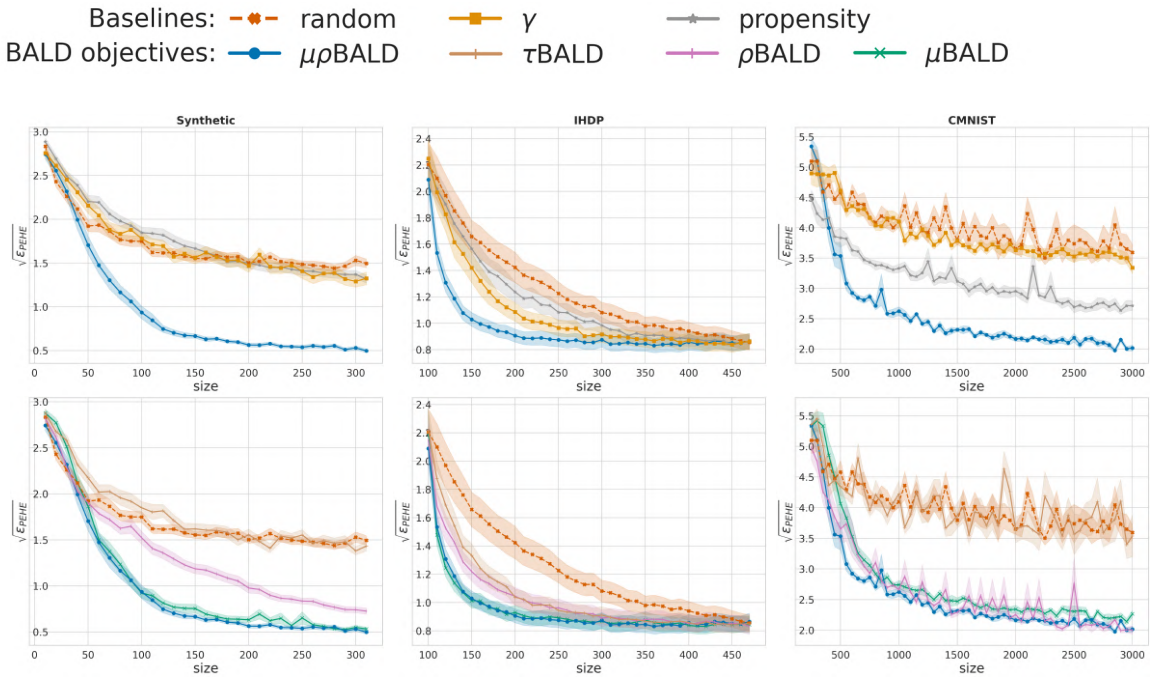


Figure A.7: $\sqrt{\epsilon_{PEHE}}$ performance (shaded standard error) for DUE models. **(left to right) synthetic** (40 seeds), and **IHDP** (200 seeds). We observe that BALD objectives outperform the **random**, γ and **propensity** acquisition functions significantly, suggesting that epistemic uncertainty aware methods that target reducible uncertainty can be more sample efficient.

A.6.6 Compute

We used a cluster of 8 nodes with 4 GPUs each (16 RTX 2080 and 16 Titan RTX). The total GPU hours is estimated to be:

8 baselines x (.5 + 1 + 1) days per dataset x (5 ensemble components * 0.25 GPU usage + 1 DUE * 0.3 GPU usage) x 24 hours = 744 GPU hours

A.6.7 Neural Network Architecture

Synthetic Architecture

```
==========================================================
Layer (type:depth-idx)                             Output Shape
==========================================================
Sequential                                         --
├─NeuralNetwork: 1-1                              [64, 100]
│    └─Sequential: 2-1                            [64, 100]
│    │    └─Linear: 3-1                           [64, 100]
│    │    └─ResidualDense: 3-2                    [64, 100]
│    │    │    └─PreactivationDense: 4-1          [64, 100]
│    │    │    │    └─Sequential: 5-1             [64, 100]
│    │    │    │    │    └─Activation: 6-1        [64, 100]
│    │    │    │    │    └─Linear: 6-2            [64, 100]
│    │    │    └─Identity: 4-2                    [64, 100]
│    │    └─ResidualDense: 3-3                    [64, 100]
│    │    │    └─PreactivationDense: 4-3          [64, 100]
│    │    │    │    └─Sequential: 5-2             [64, 100]
│    │    │    │    │    └─Activation: 6-3        [64, 100]
│    │    │    │    │    └─Linear: 6-4            [64, 100]
│    │    │    └─Identity: 4-4                    [64, 100]
│    │    └─ResidualDense: 3-4                    [64, 100]
│    │    │    └─PreactivationDense: 4-5          [64, 100]
│    │    │    │    └─Sequential: 5-3             [64, 100]
│    │    │    │    │    └─Activation: 6-5        [64, 100]
│    │    │    │    │    └─Linear: 6-6            [64, 100]
│    │    │    └─Identity: 4-6                    [64, 100]
│    │    └─Activation: 3-5                       [64, 100]
│    │    │    └─Sequential: 4-7                  [64, 100]
│    │    │    │    └─Identity: 5-4               [64, 100]
│    │    │    │    └─LeakyReLU: 5-5              [64, 100]
│    │    │    │    └─Dropout: 5-6                [64, 100]
├─GMM: 1-2                                        [64, 5]
│    └─Linear: 2-2                                [64, 5]
│    └─Linear: 2-3                                [64, 5]
│    └─Sequential: 2-4                            [64, 5]
│    │    └─Linear: 3-6                           [64, 5]
│    │    └─Softplus: 3-7                         [64, 5]
==========================================================
Total params: 32,115
```

IHDP Architecture

```
==========================================================
Layer (type:depth-idx)                             Output Shape
==========================================================
Sequential                                         --
├─TARNet: 1-1                                      [64, 400]
│    └─NeuralNetwork: 2-1                          [64, 400]
│    │    └─Sequential: 3-1                        [64, 400]
│    │    │    └─Linear: 4-1                       [64, 400]
│    │    │    └─ResidualDense: 4-2                [64, 400]
│    │    │    │    └─PreactivationDense: 5-1      [64, 400]
│    │    │    │    │    └─Sequential: 6-1         [64, 400]
│    │    │    │    └─Identity: 5-2                [64, 400]
│    │    │    └─ResidualDense: 4-3                [64, 400]
│    │    │    │    └─PreactivationDense: 5-3      [64, 400]
│    │    │    │    │    └─Sequential: 6-2         [64, 400]
│    │    │    │    └─Identity: 5-4                [64, 400]
│    └─Sequential: 2-2                             [64, 400]
│    │    └─ResidualDense: 3-2                     [64, 400]
│    │    │    └─PreactivationDense: 4-4           [64, 400]
│    │    │    │    └─Sequential: 5-5              [64, 400]
│    │    │    │    │    └─Activation: 6-3         [64, 401]
│    │    │    │    │    └─Linear: 6-4             [64, 400]
│    │    │    └─Sequential: 4-5                   [64, 400]
│    │    │    │    └─Dropout: 5-6                 [64, 401]
│    │    │    │    └─Linear: 5-7                  [64, 400]
│    │    └─ResidualDense: 3-3                     [64, 400]
│    │    │    └─PreactivationDense: 4-6           [64, 400]
│    │    │    │    └─Sequential: 5-8              [64, 400]
│    │    │    │    │    └─Activation: 6-5         [64, 400]
│    │    │    │    │    └─Linear: 6-6             [64, 400]
│    │    │    └─Identity: 4-7                     [64, 400]
│    │    └─Activation: 3-4                        [64, 400]
│    │    │    └─Sequential: 4-8                   [64, 400]
│    │    │    │    └─Identity: 5-9                [64, 400]
│    │    │    │    └─ELU: 5-10                    [64, 400]
│    │    │    │    └─Dropout: 5-11                [64, 400]
├─GMM: 1-2                                         [64, 5]
│    └─Linear: 2-3                                 [64, 5]
│    └─Linear: 2-4                                 [64, 5]
│    └─Sequential: 2-5                             [64, 5]
│    │    └─Linear: 3-5                            [64, 5]
│    │    └─Softplus: 3-6                          [64, 5]
==========================================================
```

CMNIST Architecture

```
==========================================================================
Layer (type:depth-idx)                                   Output Shape
==========================================================================
Sequential                                               --
├─TARNet: 1-1                                            [200, 100]
```

```
│    └─ResNet: 2-1                                        [200, 48]
│    │    └─Sequential: 3-1                               [200, 48, 1, 1]
│    │    │    └─Conv2d: 4-1                              [200, 12, 28, 28]
│    │    │    └─Identity: 4-2                            [200, 12, 28, 28]
│    │    │    └─ResidualConv: 4-3                        [200, 12, 28, 28]
│    │    │    │    └─Sequential: 5-1                     [200, 12, 28, 28]
│    │    │    │    │    └─PreactivationConv: 6-1         [200, 12, 28, 28]
│    │    │    │    │    └─PreactivationConv: 6-2         [200, 12, 28, 28]
│    │    │    │    └─Sequential: 5-2                     [200, 12, 28, 28]
│    │    │    │    │    └─Dropout2d: 6-3                 [200, 12, 28, 28]
│    │    │    │    │    └─Conv2d: 6-4                    [200, 12, 28, 28]
│    │    │    └─ResidualConv: 4-4                        [200, 24, 14, 14]
│    │    │    │    └─Sequential: 5-3                     [200, 24, 14, 14]
│    │    │    │    │    └─PreactivationConv: 6-5         [200, 12, 28, 28]
│    │    │    │    │    └─PreactivationConv: 6-6         [200, 24, 14, 14]
│    │    │    │    └─Sequential: 5-4                     [200, 24, 14, 14]
│    │    │    │    │    └─Dropout2d: 6-7                 [200, 12, 28, 28]
│    │    │    │    │    └─Conv2d: 6-8                    [200, 24, 14, 14]
│    │    │    └─ResidualConv: 4-5                        [200, 24, 14, 14]
│    │    │    │    └─Sequential: 5-5                     [200, 24, 14, 14]
│    │    │    │    │    └─PreactivationConv: 6-9         [200, 24, 14, 14]
│    │    │    │    │    └─PreactivationConv: 6-10        [200, 24, 14, 14]
│    │    │    │    └─Sequential: 5-6                     [200, 24, 14, 14]
│    │    │    │    │    └─Dropout2d: 6-11                [200, 24, 14, 14]
│    │    │    │    │    └─Conv2d: 6-12                   [200, 24, 14, 14]
│    │    │    └─ResidualConv: 4-6                        [200, 48, 7, 7]
│    │    │    │    └─Sequential: 5-7                     [200, 48, 7, 7]
│    │    │    │    │    └─PreactivationConv: 6-13        [200, 24, 14, 14]
│    │    │    │    │    └─PreactivationConv: 6-14        [200, 48, 7, 7]
│    │    │    │    └─Sequential: 5-8                     [200, 48, 7, 7]
│    │    │    │    │    └─Dropout2d: 6-15                [200, 24, 14, 14]
│    │    │    │    │    └─Conv2d: 6-16                   [200, 48, 7, 7]
│    │    │    └─ResidualConv: 4-7                        [200, 48, 7, 7]
│    │    │    │    └─Sequential: 5-9                     [200, 48, 7, 7]
│    │    │    │    │    └─PreactivationConv: 6-17        [200, 48, 7, 7]
│    │    │    │    │    └─PreactivationConv: 6-18        [200, 48, 7, 7]
│    │    │    │    └─Sequential: 5-10                    [200, 48, 7, 7]
│    │    │    │    │    └─Dropout2d: 6-19                [200, 48, 7, 7]
│    │    │    │    │    └─Conv2d: 6-20                   [200, 48, 7, 7]
│    │    │    └─ResidualConv: 4-8                        [200, 48, 7, 7]
│    │    │    │    └─Sequential: 5-11                    [200, 48, 7, 7]
│    │    │    │    │    └─PreactivationConv: 6-21        [200, 48, 7, 7]
│    │    │    │    │    └─PreactivationConv: 6-22        [200, 48, 7, 7]
│    │    │    │    └─Sequential: 5-12                    [200, 48, 7, 7]
│    │    │    │    │    └─Dropout2d: 6-23                [200, 48, 7, 7]
│    │    │    │    │    └─Conv2d: 6-24                   [200, 48, 7, 7]
│    │    │    └─AdaptiveAvgPool2d: 4-9                   [200, 48, 1, 1]
│    └─Sequential: 2-2                                    [200, 100]
│    │    └─ResidualDense: 3-2                            [200, 100]
│    │    │    └─PreactivationDense: 4-10                 [200, 100]
```

```
│    │    │    │    └─Sequential: 5-13                  [200, 100]
│    │    │    │    │    └─Activation: 6-25             [200, 49]
│    │    │    │    │    └─Linear: 6-26                 [200, 100]
│    │    │    └─Sequential: 4-11                       [200, 100]
│    │    │    │    └─Dropout: 5-14                     [200, 49]
│    │    │    │    └─Linear: 5-15                      [200, 100]
│    │    └─ResidualDense: 3-3                          [200, 100]
│    │    │    └─PreactivationDense: 4-12               [200, 100]
│    │    │    │    └─Sequential: 5-16                  [200, 100]
│    │    │    │    │    └─Activation: 6-27             [200, 100]
│    │    │    │    │    └─Linear: 6-28                 [200, 100]
│    │    │    └─Identity: 4-13                         [200, 100]
│    │    └─Activation: 3-4                             [200, 100]
│    │    │    └─Sequential: 4-14                       [200, 100]
│    │    │    │    └─Identity: 5-17                    [200, 100]
│    │    │    │    └─LeakyReLU: 5-18                   [200, 100]
│    │    │    │    └─Dropout: 5-19                     [200, 100]
├─GMM: 1-2                                              [200, 5]
│    └─Linear: 2-3                                      [200, 5]
│    └─Linear: 2-4                                      [200, 5]
│    └─Sequential: 2-5                                  [200, 5]
│    │    └─Linear: 3-5                                 [200, 5]
│    │    └─Softplus: 3-6                               [200, 5]
==========================================================================
```

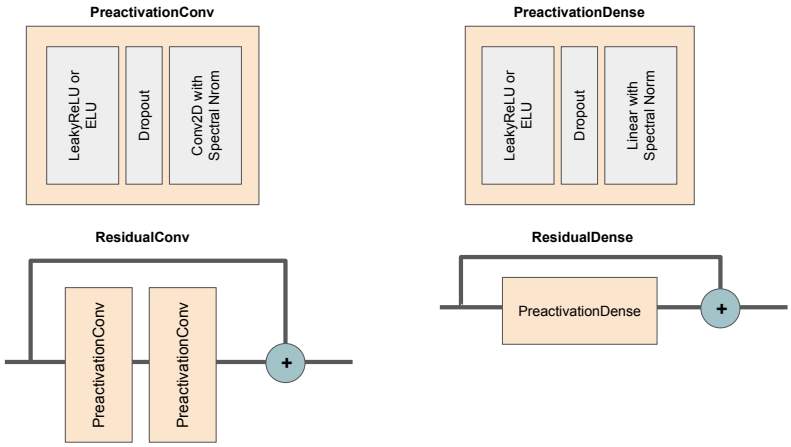


Figure A.8: `PreactivationConv` is a convolution layer with LeakyReLU (or ELU when slope is negative) activation, dropout and spectral norm applied [Gouk et al., 2021; Miyato et al., 2018]. Similarly, `PreactivationDense` is a dense layer with BatchNorm [Ioffe and Szegedy, 2015], LeakyReLU (or ELU when slope is negative) activation and spectral norm applied [Gouk et al., 2021; Miyato et al., 2018]. `ResidualConv` is the residual convolution layer, defined as `PreactivationConv(PreactivationConv(x)) + SpectralNorm(1x1Conv(x))` and `ResidualDense` are residual dense layers, defined as `PreactivationDense(x)+x)`.

All experiments were trained using Adam optimizer.

A.6.7.1 Hyper-Parameters

Table A.2: Training hyper parameters for **Deep Ensemble** experiments

Parameter	**Synthetic**	**IHDP**	**CMNIST**
dim hidden	100	400	100
dropout	0.0	0.15	0.1
depth	4	3	3
spectral norm	12	0.95	24
learning rate	0.001	0.001	0.001
negative slope	0.0	−1.0	0.0

Table A.3: Training hyper parameters for **DUE** experiments

Parameter	**Synthetic**	**IHDP**	**CMNIST**
inducing points	100	100	100
dim hidden	100	200	200
dropout	0.2	0.1	0.05
depth	3	3	2
batch size	200	100	64
spectral norm	0.95	0.95	3.0
learning rate	0.001	0.001	0.001
negative slope	0.0	−1.0	−1.0

The first principle is that you must not fool yourself—and you are the easiest person to fool.

Richard Feynman

B Reproducibility Analysis

B.1 Deep Learning on a Data Diet

The senior author of 'Deep Learning on a Data Diet' [Paul et al., 2021] recently gave a talk at our lab that explored this issue, presenting their novel metrics for pruning datasets. During the talk, the author of this current work suggested *a correlation between the proposed GraNd score at initialization and input norms*, sparking further research into the effectiveness of these new pruning techniques. In this chapter, we delve deeper into this intriguing question, exploring the practicality and efficacy of these metrics for data pruning.

'Deep Learning on a Data Diet'. Paul et al. [2021] introduce two novel metrics: *Error L2 Norm (EL2N)* and *Gradient Norm at Initialization (GraNd)*. These metrics aim to provide a more effective means of dataset pruning. It is important to emphasize that the GraNd score at initialization is calculated before any training has taken place, averaging over several randomly initialized models. This fact has been met with skepticism by reviewers[1], but Paul et al. [2021] specifically remark on GraNd at initialization:

> **Pruning at initialization.** In all settings, GraNd scores can be used to select a training subset at initialization that achieves test accuracy significantly better than random, and in some cases, competitive with training on all the data. This is remarkable because GraNd only contains information about the gradient norm at initialization. This suggests that the geometry of the training distribution induced by a random network contains a surprising amount of information about the structure of the classification problem.

GraNd. The GraNd score measures the magnitude of the gradient vector for a specific input sample in the context of neural network training over different parameter draws. The formula for calculating the (expected) gradient norm is:

$$\text{GraNd}(x) = \mathbb{E}_{\theta_t}[\|\nabla_{\theta_t} L(f(x;\theta_t), y)\|_2] \tag{B.1}$$

where $\nabla_{\theta_t} L(f(\boldsymbol{x};\theta_t), y)$ is the gradient of the loss function L with respect to the model's parameters θ_t at epoch t, $f(\boldsymbol{x};\theta)$ is the model's prediction for input $\boldsymbol{x}$, and y is the true label for the input. We take an expectation over several training runs. The gradient norm provides information about the model's sensitivity to a particular input and helps in identifying data points that have a strong influence on the learning process.

[1] See also https://openreview.net/forum?id=Uj7pF-D-YvT&noteId=qwy3HouKSX.

EL2N. The EL2N score measures the squared difference between the predicted and (one-hot) true labels for a specific input sample. The formula for calculating the EL2N score is:

$$\text{EL2N}(x) = \mathbb{E}_{\theta_t}[\|f(x;\theta_t) - y\|_2^2] \tag{B.2}$$

where $f(\boldsymbol{x};\theta)$ is the model's prediction for input $\boldsymbol{x}$, y is the (one-hot) true label for the input, and $\|\cdot\|_2$ denotes the Euclidean (L2) norm. The EL2N score provides insight into the model's performance on individual data points, allowing for a more targeted analysis of errors and potential improvements.

The GraNd and EL2N scores are proposed in the context of dataset pruning, where the goal is to remove less informative samples from the training data. Thus, one can create a smaller, more efficient dataset that maintains the model's overall performance while reducing training time and computational resources.

While GraNd at initialization does not require model training, it requires a model and is not cheap to compute. In contrast, the input norm of training samples is incredibly cheap to compute and would thus provide an exciting new baseline to use for data pruning experiments. We investigate this correlation in this chapter and find positive evidence for it. However, we also find that the GraNd score at initialization does not outperform random pruning, unlike the respective results of Paul et al. [2021] for GraNd at initialization.

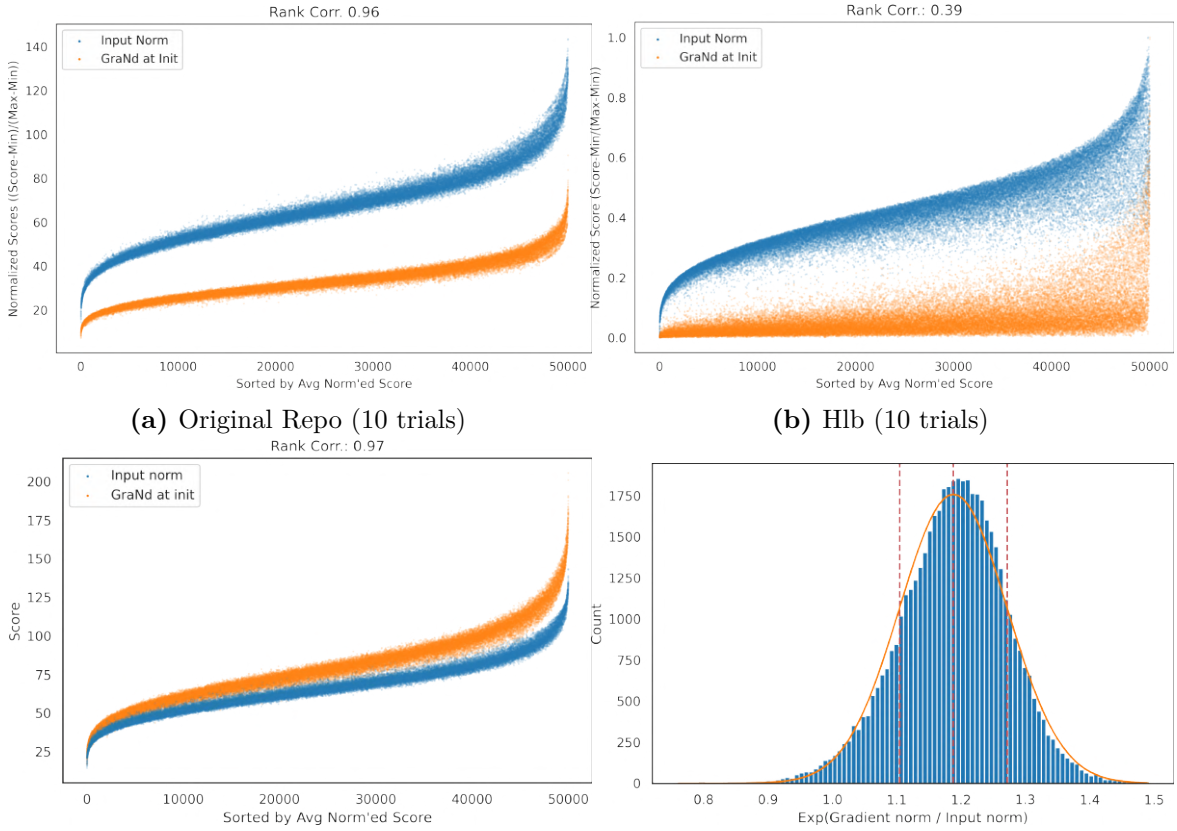
Outline. In §B.1.1.1, we begin by discussing the correlation between input norm and gradient norm at initialization. We empirically find strong correlation between GraNd scores at initialization and input norms as we average over models. In §B.1.1.2, we explore the implication of this insight for dataset pruning and find that both GraNd at initialization and input norm scores do not outperform random pruning, but GraNd scores after a few epochs perform similar to EL2N scores at these later epochs.

In summary, this reproduction contributes a new insight on the relationship between input norm and gradient norm at initialization and finds a failure to reproduce one of the six contributions of Paul et al. [2021].

B.1.1 Investigation

We investigate the correlation between input norm and GraNd at initialization and the other scores on CIFAR-10 [Krizhevsky, 2009] in three different ways: First, we update the original paper repository[2] (`https://github.com/mansheej/data_diet`), which uses JAX [Bradbury et al., 2018], rerun the experiments for Figure 1 (second row) in Paul et al. [2021] for CIFAR-10, which trains for 200 epochs, using GraNd at initialization, GraNd at epoch 20, E2LN at epoch 20, Forget Score at epoch 200, and input norm. Second, we reproduce the same experiments using 'hlb' [Balsam, 2023], which is a strongly modified version of ResNet-18 that allows to train to high accuracy in 12 epochs taking about 30 seconds total on an Nvidia RTX 4090 in PyTorch [Paszke et al., 2019] For the latter, we compare GraNd at initialization, GraNd at epoch 1 ($\approx 20/200 \cdot 12$ epochs), EL2N at epoch 1, and input norm[3]. Third, we compare the rank correlations between the different scores for those two repositories and also use another 'minimal' CIFAR-10 implementation [van Amersfoort, 2021] with a standard ResNet18 architecture for CIFAR-10 to compare the rank correlations.

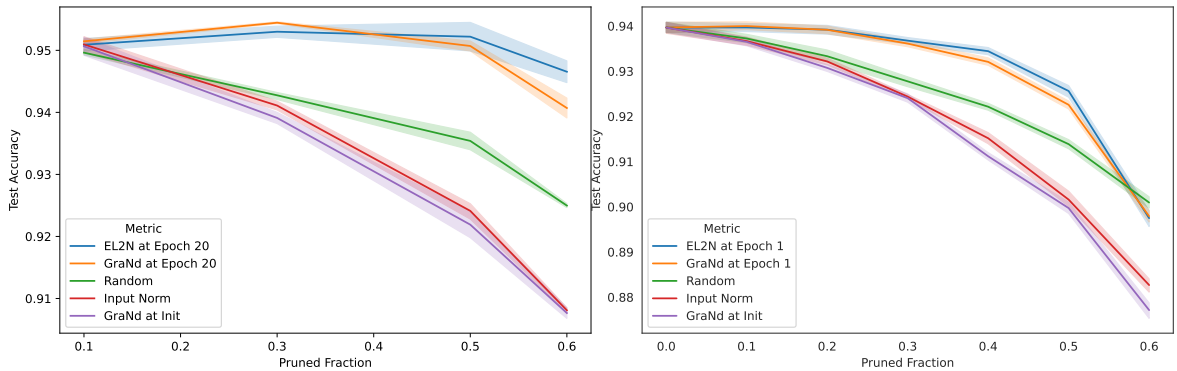
[2] `https://github.com/blackhc/data_diet`
[3] `https://github.com/blackhc/pytorch_datadiet`



(a) Original Repo (10 trials) **(b)** Hlb (10 trials)

(c) Minimal (120 trials) **(d)** Minimal (120 trials)

Figure B.1: *Correlation between GraNd at Initialization and Input Norm for CIFAR-10's training set.* **(a)**, **(b)**, **(c)**: We sort the samples by their average normalized score (i.e., the score minus its minimum divided by its range), plot the scores and compute Spearman's rank correlation on CIFAR-10's training data. The original repository and the 'minimal' implementation have very high rank correlation—'hlb' has a lower but still strong rank correlation. **(d)**: *Ratio between input norm and gradient norm.* In the 'minimal' implementation, the ratio between input norm and gradient norm is roughly log-normal distributed.



(a) Original Repo (2 trials each) **(b)** Hlb (10 trials)

Figure B.2: *Reproduction of Figure 1 (second row) from Paul et al. [2021].* In both reproductions, GraNd at initialization performs as well as the input norm. However, it does not perform better than random pruning. Importantly, it also fails to reproduce the results from Paul et al. [2021]. However, GraNd at epoch 20 (respectively at epoch 1 for 'hlb') performs similar to EL2N and like GraNd at initialization in Paul et al. [2021].

B.1.1.1 Correlation between GraNd at Initialization and Input Norm

To better understand the relationship between the input norm and the gradient norm at initialization, let us consider a toy example first and then appeal to empirical evidence as is common in deep learning research: let's examine linear softmax classification with C classes (without a bias term). The model takes the form:

$$f(x) = \text{softmax}(Wx), \tag{B.3}$$

together with the cross-entropy loss function:

$$L = -\log f(x)_y. \tag{B.4}$$

The gradient of the loss function with respect to the rows w_j of the weight matrix W is:

$$\nabla_{w_j} L = (f(x)_j - \mathbb{1}\{j = y\})x \tag{B.5}$$

where $\mathbb{1}\{j = y\}$ is the indicator function that is 1 if $j = y$ and 0 otherwise. The squared norm of the gradient is:

$$\|\nabla_w L\|_2^2 = \sum_{j=1}^{C} (f(x)_j - \mathbb{1}\{j = y\})^2 \|x\|_2^2. \tag{B.6}$$

In expectation over W (different initializations), the norm of the gradient is:

$$\mathbb{E}_W \left[\|\nabla_w L\|_2\right] = \mathbb{E}_W \left[\left(\sum_{j=1}^{C} (f(x)_j - \mathbb{1}\{j = y\})^2 \right)^{1/2} \right] \|x\|_2. \tag{B.7}$$

Thus, we see that the gradient norm is a multiple of the input norm. The factor depends on $f(\boldsymbol{x})_j$, which we could typically expect to be $1/C$.

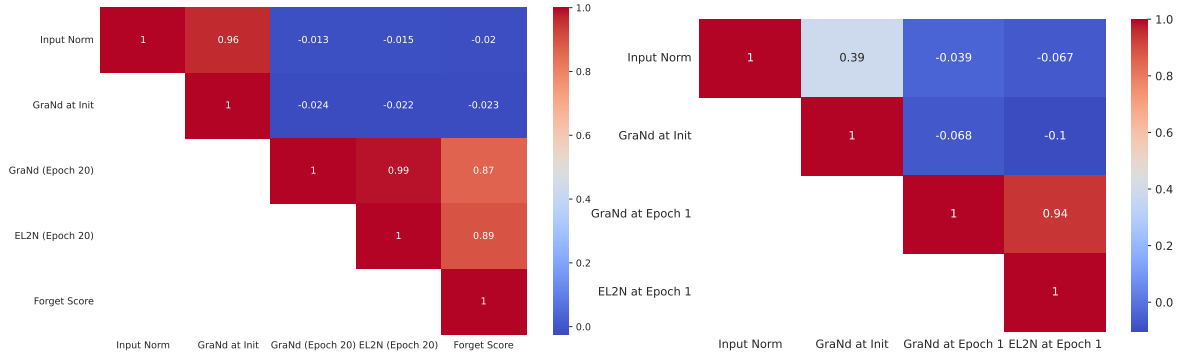
Empirical Evidence. In Figure B.1, we see that on CIFAR-10's training set, GraNd at initialization and the input norm are highly correlated. This is true for the original repository, the 'hlb' and the 'minimal' implementation. The 'hlb' implementation has a lower but still strong correlation.

B.1.1.2 Reproducing Figure 1 of Paul et al. [2021] on CIFAR-10

In Figure B.2, we see that GraNd at initialization performs about as well as using the input norm. However, it does not reproduce the results from Paul et al. [2021]. It performs worse than random pruning (for 'hlb'). However, GraNd at epoch 20 (respectively at epoch 1 for 'hlb') performs like GraNd at initialization in Paul et al. [2021]. Similarly, in Figure B.3, we see that GraNd at initialization and the input norm are strongly correlated as are GraNd at later epochs, EL2N and the Forget Score, with little correlation between these two groups.

B.1.2 Discussion

If GraNd at initialization performed as well as claimed in Paul et al. [2021], using the input norm would provide a new exciting baseline for data pruning because it is model independent and cheaper to compute than GraNd or other scores. However, since only GraNd at later epochs seems to perform as expected, we cannot recommend using input norm or GraNd at initialization for data pruning.



(a) Original Repo (2 trials each) **(b)** Hlb (10 trials)

Figure B.3: *Rank Correlations of the Scores.* Cf. Figure 12 in the appendix of Paul et al. [2021]. In both reproductions, GraNd at initialization and input norm are positively correlated, while GraNd and EL2N at later epochs are strongly correlated with each other and the Forget Score (at epoch 200).

As to the failure to reproduce the results of Paul et al. [2021], we could not rerun the code using the original JAX version because it is too old for our GPU. The authors of Paul et al. [2021] were, however, able to set up a Google Cloud VM with an old image that was able to reproduce the original results using the original JAX version. On further investigation, the author of this reproduction found a bug in `flax.training.restore_checkpoint` that was fixed in April 2021[4]: passing a `0` step (i.e. initialization) would trigger loading the *latest* checkpoint instead of the zeroth checkpoint because the internal implementation was checking `if step:` instead of `if step is not None:` when deciding whether to fall back to loading the latest checkpoint. This bug was fixed in April 2021, but the authors of Paul et al. [2021] were not aware of this bug and did not rerun their experiments with newer JAX/FLAX versions. We have accordingly informed the authors of Paul et al. [2021].

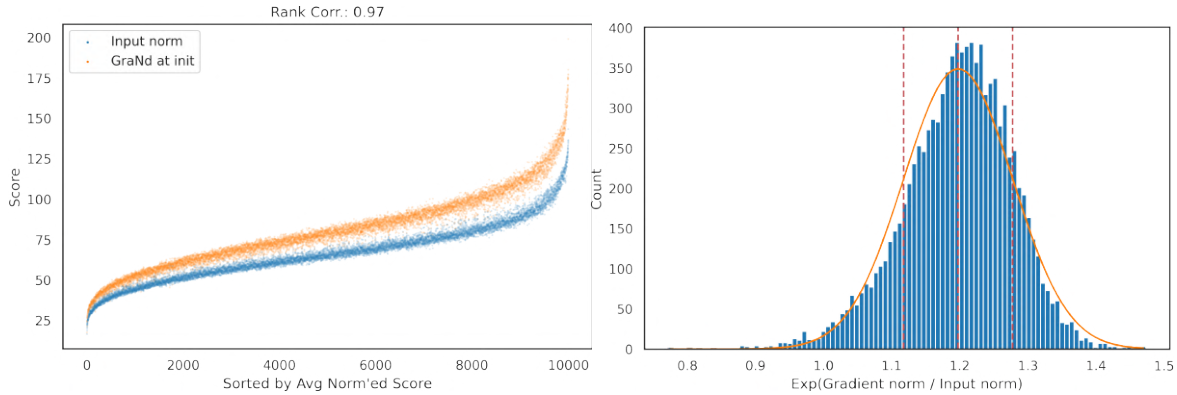
B.1.3 Details

B.2 A Note on "Assessing Generalization of SGD via Disagreement"

Machine learning models can cause harm when their predictions become unreliable, yet we trust them blindly. This is not fiction but has happened in real-world applications of machine learning [Schneider, 2021]. Thus, there has been significant research interest in model robustness, uncertainty quantification, and bias mitigation. In particular, finding ways to bound the test error of a trained deep neural network without access to the labels would be of great importance: one could estimate the performance of models in the wild where unlabeled data is ubiquitous, labeling is expensive, and the data often does not match the training distribution. Crucially, it would provide a signal on when to trust the output of a model and when to defer to human experts instead.

Several recent works [Chen et al., 2021; Granese et al., 2021; Garg et al., 2022; Jiang et al., 2022] look at the question of how model predictions in a non-Bayesian setting can be used to estimate model accuracy. In this chapter, we focus on one

[4]See https://github.com/google/flax/commit/28fbd95500f4bf2f9924d2560062fa50e919b1a5.



(a) Minimal (1000 trials) **(b)** Minimal (1000 trials)

Figure B.4: *Correlation between GraNd at Initialization and Input Norm on the Test Set.* **((a))**: We sort the samples by their average normalized score (i.e., the score minus its minimum divided by its range), plot the scores and compute Spearman's rank correlation on CIFAR-10's test data. The original repository and the 'minimal' implementation have very high rank correlation—'hlb' has a lower but still strong rank correlation. **((b))**: *Ratio between input norm and gradient norm.* In the 'minimal' implementation, the ratio between input norm and gradient norm is roughly log-normal distributed

work[5] [Jiang et al., 2022] specifically, and examine the theoretical and empirical results from a Bayesian perspective.

As we have already examined in §1.2.3 and §3, *epistemic uncertainty* [Der Kiureghian and Ditlevsen, 2009] captures the uncertainty of a model about the reliability of its predictions, that is epistemic uncertainty quantifies the uncertainty of a model about its predictive distribution, while *aleatoric uncertainty* quantifies the ambiguity within the predictive distribution and the label noise, c.f. Kendall and Gal [2017]. Epistemic uncertainty thus tells us whether we can trust a model's predictions or not. Assuming a *well-specified* and *well-calibrated* Bayesian model, when its predictive distribution has low epistemic uncertainty for an input, it can be trusted. But likewise, a Bayesian model's calibration ought to deteriorate as epistemic uncertainty increases for a sample: in that case, the predictions become less reliable and so does the model's calibration.

In this context, calibration is an *aleatoric* metric for a model's reliability [Gopal, 2021]. Calibration captures how well a model's confidence for a given prediction matches the actual frequency of that prediction in the limit of observations *in distribution*: when a model is 70% confident about assigning label A, does A indeed occur with 70% probability in these instances?

Jiang et al. [2022], while not Bayesian, make the very interesting empirical and theoretical discovery that deep ensembles satisfy a '*Generalization Disagreement Equality*' when they are well-calibrated according to a proposed '*class-aggregated calibration*' (or a '*class-wise calibration*') and empirically find that the respective calibration error generally bounds the absolute difference between the test error and '*disagreement rate*.' Jiang et al. [2022]'s theory builds upon Nakkiran and Bansal [2020]'s '*Agreement Property*' and provides backing for an empirical connection between the *test error* and *disagreement rate* of two separately trained networks on the same training data. Yet, while Nakkiran and Bansal [2020] limit the applicability of their

[5] An ICLR 2022 Spotlight, which has spawned additional follow-up works, e.g. Baek et al. [2022].

Agreement Property to in-distribution data, Jiang et al. [2022] carefully extend it: '*our theory is general and makes no restrictions on the hypothesis class, the algorithm, the source of stochasticity, or the test distributions (which may be different from the training distribution)*' with qualified evidence: '*we present preliminary observations showing that GDE is approximately satisfied even for certain distribution shifts within the PACS [Li et al., 2017] dataset.*'

In this chapter, we present a new perspective on the theoretical results using a standard probabilistic approach for discriminative (Bayesian) models, whereas Jiang et al. [2022] use a hypothesis space of models that output one-hot predictions. Indeed, their theory does not require one-hot predictions, separately trained models (deep ensembles) or Bayesian models. Moreover, as remarked by the authors, their theoretical results also apply to a single model that outputs softmax probabilities. We will see that our perspective greatly simplifies the results and proofs.

This also means that the employed notion of disagreement rate *does not capture epistemic uncertainty but overall uncertainty*, similar to the predictive entropy, which is a major difference to Bayesian approaches which can evaluate epistemic uncertainty separately [Smith and Gal, 2018]. Overall uncertainty is the sum of aleatoric and epistemic uncertainty. For in-distribution data, epistemic uncertainty will generally be low: overall uncertainty will mainly capture aleatoric uncertainty and align well with (aleatoric) calibration measures. However, under distribution shift, epistemic uncertainty can be a confounding factor.

Importantly, we find that the connection between the proposed calibration metrics and the gap between test error and disagreement rate exists because the introduced notion of class-aggregated calibration is so strong that this connection follows almost at once.

Moreover, the suggested approach is circular[6]: calibration must be measured on the data distribution we want to evaluate. Otherwise, we cannot bound the difference between the test error and the disagreement rate and obtain a signal on how trustworthy our model is. This reintroduces the need for labels on the unlabeled dataset, limiting practicality. Alternatively, one would have to assume that these calibration metrics do not change for different datasets or under distribution shifts, which we show not to hold: deep ensembles are less calibrated the more the ensemble members disagree (even on in-distribution data).

Lastly, we draw connections and show that the 'class-aggregated calibration error' and the 'class-wise calibration error'[7] are equivalent to the 'adaptive calibration error' and 'static calibration error' introduced in Nixon et al. [2019] and its implementation.

Outline.

We introduce the necessary background and notation in §B.2.1. In §B.2.2 we rephrase the theoretical statements from Jiang et al. [2022] using a parameter distribution (instead of a version space) and auxiliary random variables. This allows us to simplify the theoretical statements and proofs greatly in §B.2.3 and to examine the connection to Nixon et al. [2019]. Finally, in §B.2.4, we provide empirical evidence that deep ensembles are less calibrated exactly when their ensemble members disagree.

[6]This was added as a caveat to the camera-ready version of Jiang et al. [2022] after reviewing a preprint of the preprint of the paper this chapter is based on.

[7]Which is not explicitly introduced in Jiang et al. [2022] but can be analogously constructed.

B.2.1 Background & Setting

In this section, we introduce additional notation, the initial Bayesian formalism, the connection to deep ensembles, and the probabilistic model. We restate the statements from Jiang et al. [2022] using this formalism in §B.2.2.

Notation. We use an implicit notation for expectations $\mathbb{E}[f(X)]$ when possible. For additional clarity, we also use $\mathbb{E}_X[f(X)]$ and $\mathbb{E}_{\mathrm{p}(x)}\, f(x)$, which fix the random variables and distribution, respectively, when needed.

We will use nested probabilistic expressions of the form $\mathbb{E}[\mathrm{p}(\hat{Y} = Y \mid X)]$. Prima facie, this seems unambiguous, but is $\mathrm{p}(\hat{Y} = Y \mid X)$ a transformed random variable of only X or also of Y (and $\hat{Y}$): what are we taking the expectation over? This is not always unambiguous, so we disambiguate between the probability for an event defined by an expression $\mathbb{P}[\ldots] = \mathbb{E}[\mathbb{1}\{\ldots\}]$, where $\mathbb{1}\{\ldots\}$ is the indicator function[8], and a probability given specific outcomes for various random variables $\mathrm{p}(\hat{y} \mid x)$, c.f.:

$$\mathbb{P}[\hat{Y} = Y \mid X] = \mathbb{E}_{\hat{Y},Y}[\mathbb{1}\{\hat{Y} = Y\} \mid X] = \mathbb{E}_{\mathrm{p}(\hat{y},y|X)}\, \mathbb{1}\{\hat{y} = y\}, \tag{B.8}$$

which is a transformed random variable of X, while $\mathrm{p}(\hat{Y} = Y \mid X)$ is simply a (transformed) random variable, applying the probability density on the random variables Y and X. Put differently, Y is bound within the former but not the latter: $\mathbb{P}[\ldots \mid X]$ is a transformed random variable of X, and any random variable that appears within the $\ldots$ is bound within that expression.

Probabilistic Model. We assume classification with $\mathtt{C}$ classes. For inputs X with ground-truth labels Y, we have a Bayesian model with parameters Ω that makes predictions $\hat{Y}$:

$$\mathrm{p}(y, \hat{y}, \omega \mid x) = \mathrm{p}(y \mid x)\ \mathrm{p}(\hat{y} \mid x, \omega)\ \mathrm{p}(\omega). \tag{B.9}$$

We focus on model evaluation. (Input) samples x can come either from 'in-distribution data' which follows the training set or from samples under covariate shift (distribution shift). The expected prediction over the model parameters is the *marginal predictive distribution*:

$$\mathrm{p}(\hat{y} \mid x) = \mathbb{E}_{\Omega}[\mathrm{p}(\hat{y} \mid x, \Omega)]. \tag{B.10}$$

On $\mathrm{p}(\omega)$. The main emphasis in Bayesian modelling can be Bayesian inference or Bayesian model averaging [Wilson and Izmailov, 2020]. Here we concentrate on the model averaging perspective, and for simplicity take the model averaging to be with respect to *some* distribution $\mathrm{p}(\omega)$. Hence, we will use $\mathrm{p}(\omega)$ as the push-forward of models initialized with different initial seeds through SGD to minimize the negative log likelihood with weight decay and a specific learning rate schedule (MLE or MAP)—the same we also did in §3:

Assumption B.1. We assume that $\mathrm{p}(\omega)$ is a distribution of possible models we obtain by training with a specific training regime on the training data with different seeds. A single ω identifies a single trained model.

We cast deep ensembles [Hansen and Salamon, 1990; Lakshminarayanan et al., 2017], which refer to training multiple models and averaging predictions, into the

[8] The indicator function is 1 when the predicate '…' is true and 0 otherwise.

introduced Bayesian perspective above by viewing them as an empirical finite sample estimate of the parameter distribution $\mathrm{p}(\omega)$. Then, $\omega_1, \ldots, \omega_N \sim \mathrm{p}(\omega)$ drawn i.i.d. are the *ensemble members*.

Again, the implicit model parameter distribution $\mathrm{p}(w)$ is given by the models that are obtained through training. Hence, we can view the predictions of a deep ensemble or the ensemble's prediction disagreement for specific x (or over the data) as empirical estimates of the predictions or the model disagreement using the implicit model distribution, respectively.

Calibration. A model's calibration for a given x measures how well the model's *top-1 (argmax) confidence*

$$\mathrm{Conf}_{\mathrm{Top1}} \triangleq \mathrm{p}(\hat{Y} = \arg\max_k \mathrm{p}(\hat{Y} = k \mid X) \mid X) \tag{B.11}$$

matches its *top-1 accuracy*

$$\mathrm{Acc}_{\mathrm{Top1}} \triangleq \mathrm{p}(Y = \arg\max_k \mathrm{p}(\hat{Y} = k \mid X) \mid X), \tag{B.12}$$

where we define both as transformed random variables of X. The calibration error is usually defined as the absolute difference between the two:

$$\mathrm{CE} \triangleq |\mathrm{Acc}_{\mathrm{Top1}} - \mathrm{Conf}_{\mathrm{Top1}}|. \tag{B.13}$$

In general, we are interested in the *expected calibration error (ECE)* over the data distribution [Guo et al., 2017] where we bin samples by their top-1 confidence. Intuitively, the ECE will be low when we can trust the model's top-1 confidence on the given data distribution.

We usually use top-1 predictions in machine learning. However, if we were to draw $\hat{Y}$ according to $\mathrm{p}(\hat{y} \mid x)$ instead, the (expected) accuracy would be:

$$\begin{aligned} \mathrm{Acc} &\triangleq \mathbb{P}[Y = \hat{Y} \mid X] && \text{(B.14)} \\ &= \sum_k \mathrm{p}(Y = k \mid X)\ \mathrm{p}(\hat{Y} = k \mid X) && \text{(B.15)} \\ &= \mathbb{E}_Y[\mathrm{p}(\hat{Y} = Y \mid X) \mid X], && \text{(B.16)} \end{aligned}$$

as a random variable of X. Usually we are interested in the accuracy over the whole dataset:

$$\mathbb{P}[\hat{Y} = Y] = \mathbb{E}[\mathrm{Acc}] = \mathbb{E}_X[\mathbb{P}[\hat{Y} = Y \mid X]] = \mathbb{E}_{X,Y}[\mathrm{p}(\hat{Y} = Y \mid X)]. \tag{B.17}$$

For example, for binary classification with two classes A and B, if class A appears with probability 0.7 and a model predicts class A with probability 0.2 (and thus class B appears with probability 0.3, which a model predicts as 0.8), its accuracy is $0.7 \times 0.2 + 0.3 \times 0.8 = 0.38$, while the top-1 accuracy is 0.3. Likewise, the predicted accuracy is $0.2^2 + 0.8^2 = 0.68$ while the top-1 predicted accuracy is 0.8.

B.2.2 Rephrasing Jiang et al. [2022] in a Probabilistic Context

We present the same theoretical results as Jiang et al. [2022] but use a Bayesian formulation instead of a hypothesis space and define the relevant quantities as (transformed) random variables. As such, our definitions and theorems are equivalent and follow the

paper but look different. We show these equivalences in §B.2.7.1 in the appendix and prove the theorems and statements themselves in the next section.

First, however, we note a distinctive property of Jiang et al. [2022]. It is assumed that each $\mathrm{p}(\hat{y} \mid x, \omega)$ is always one-hot for any ω. In practice, this could be achieved by turning a neural network's softmax probabilities into a one-hot prediction for the arg max class. We call this the *Top1-Output-Property* (TOP).

Assumption B.2. The Bayesian model $\mathrm{p}(\hat{y}, \omega \mid x)$ satisfies TOP: $\mathrm{p}(\hat{y} \mid x, \omega)$ is one-hot for all x and ω.

Definition B.1. The *test error* and *disagreement rate*, as transformed random variables of Ω (and Ω'), are:

$$\begin{aligned}
\text{TestError} &\triangleq \mathbb{P}[\hat{Y} \neq Y \mid \Omega] && \text{(B.18)}\\
\Big(&= 1 - \mathbb{P}[\hat{Y} = Y \mid \Omega] && \text{(B.19)}\\
&= 1 - \mathbb{E}_{X,Y}[\mathrm{p}(\hat{Y} = Y \mid X, \Omega)], && \text{(B.20)}\\
&= 1 - \mathbb{E}_{\mathrm{p}(x,y)}\, \mathrm{p}(\hat{Y} = y \mid x, \Omega)) \Big), && \text{(B.21)}\\
\text{Dis} &\triangleq \mathbb{P}[\hat{Y} \neq \hat{Y}' \mid \Omega, \Omega'] && \text{(B.22)}\\
\Big(&= 1 - \mathbb{P}[\hat{Y} = \hat{Y}' \mid \Omega, \Omega'] && \text{(B.23)}\\
&= 1 - \mathbb{E}_{X,\hat{Y}}[\mathrm{p}(\hat{Y}' = \hat{Y} \mid X, \Omega') \mid \Omega, \Omega'] && \text{(B.24)}\\
&= 1 - \mathbb{E}_{\mathrm{p}(x,\hat{y}\mid\Omega)}\, \mathrm{p}(\hat{Y}' = \hat{y} \mid x, \Omega') \Big), && \text{(B.25)}
\end{aligned}$$

where for the disagreement rate, we expand our probabilistic model to take a second model Ω' with prediction $\hat{Y}'$ into account (and which uses the same parameter distribution), so:

$$\mathrm{p}(y, \hat{y}, \omega, \hat{y}', \omega' \mid x) \triangleq \mathrm{p}(y \mid x)\ \mathrm{p}(\hat{y} \mid x, \omega)\ \mathrm{p}(\omega)\, \mathrm{p}(\hat{Y} = \hat{y}' \mid x, \Omega = \omega')\ \mathrm{p}(\Omega = \omega').$$

Jiang et al. [2022] then introduce the property of interest:

Definition B.2. A Bayesian model $\mathrm{p}(\hat{y}, \omega \mid x)$ fulfills the *Generalization Disagreement Equality (GDE)* when:

$$\mathbb{E}_{\Omega}[\text{TestError}(\Omega)] = \mathbb{E}_{\Omega,\Omega'}[\text{Dis}(\Omega, \Omega')] \quad (\Leftrightarrow \mathbb{E}[\text{TestError}] = \mathbb{E}[\text{Dis}]). \tag{B.26}$$

When this property holds, we seemingly do not require knowledge of the labels to estimate the (expected) test error: computing the (expected) disagreement rate is sufficient.

Two different types of calibration are then introduced, *class-wise* and *class-aggregated* calibration, and it is shown that they imply the GDE:

Definition B.3. The Bayesian model $\mathrm{p}(\hat{y}, \omega \mid x)$ satisfies *class-wise calibration* when for any $q \in [0, 1]$ and any class $k \in [\mathtt{C}]$:

$$\mathrm{p}(Y = k \mid \mathrm{p}(\hat{Y} = k \mid X) = q) = q. \tag{B.27}$$

Similarly, the Bayesian model $\mathrm{p}(\hat{y}, \omega \mid x)$ satisfies *class-aggregated calibration* when for any $q \in [0, 1]$:

$$\sum_k \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) = q \sum_k \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q). \tag{B.28}$$

Theorem B.1. *When the Bayesian model* $\mathrm{p}(\hat{y}, \omega \mid x)$ *satisfies class-wise or class-aggregated calibration, it also satisfies GDE.*

Finally, Jiang et al. [2022] introduce the *class-aggregated calibration error* similar to the ECE and then use it to bound the magnitude of any GDE gap:

Definition B.4. The *class-aggregated calibration error (CACE)* is the integral of the absolute difference of the two sides in Equation B.28 over possible $q \in [0, 1]$:

$$\mathrm{CACE} \triangleq \int_{q\in[0,1]} \Big| \sum_k \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) - q \sum_k \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q) \Big| dq. \tag{B.29}$$

Theorem B.2. *For any Bayesian model* $\mathrm{p}(\hat{y}, \omega \mid x)$*, we have:*

$$|\mathbb{E}[\mathrm{TestError}] - \mathbb{E}[\mathrm{Dis}]| \leq \mathrm{CACE}.$$

In the following section, we simplify the definitions and prove the statements using elementary probability theory, showing that notational complexity is the main source of complexity.

B.2.3 GDE is Class-Aggregated Calibration in Expectation

We show that proof for Theorem B.2 is trivial if we use different but equivalent definitions of the class-wise and class-aggregate calibration. First though, we establish a better understanding for these definitions by examining the GDE property $\mathbb{E}[\mathrm{TestError}] = \mathbb{E}[\mathrm{Dis}]$. For this, we expand the definitions of $\mathbb{E}[\mathrm{TestError}]$ and $\mathbb{E}[\mathrm{Dis}]$, and use random variables to our advantage.

We define a quantity which will be of intuitive use later on: the *predicted accuracy*

$$\mathrm{PredAcc} \triangleq \mathbb{E}_{\hat{Y}}[\mathrm{p}(\hat{Y} \mid X) \mid X] = \sum_k \mathrm{p}(\hat{Y} = k \mid X)\, \mathrm{p}(\hat{Y} = k \mid X), \tag{B.30}$$

as a random variable of X. It measures the expected accuracy assuming the model's predictions are correct, that is the true labels follow $\mathrm{p}(\hat{y} \mid x)$. This also assumes that we draw $\hat{Y}$ accordingly and do not always use the top-1 prediction.

Revisiting GDE. On the one hand, we have:

$$\begin{aligned}
\mathbb{E}[\mathrm{TestError}] &= \mathbb{E}_{\Omega}[\mathbb{P}[\hat{Y} \neq Y \mid \Omega]] && \text{(B.31)}\\
&= 1 - \mathbb{P}[\hat{Y} = Y] && \text{(B.32)}\\
&= 1 - \mathbb{E}_{X,\hat{Y}}[\mathrm{p}(Y = \hat{Y} \mid X)] && \text{(B.33)}\\
&= 1 - \mathbb{E}[\mathrm{Acc}] && \text{(B.34)}
\end{aligned}$$

and on the other hand, we have:

$$\begin{aligned}
\mathbb{E}[\mathrm{Dis}] &= \mathbb{E}_{\Omega,\Omega'}[\mathbb{P}[\hat{Y} \neq \hat{Y}' \mid \Omega, \Omega']] && \text{(B.35)}\\
&= 1 - \mathbb{E}_{\Omega,\Omega'}[\mathbb{P}[\hat{Y} = \hat{Y}' \mid \Omega, \Omega']] && \text{(B.36)}\\
&= 1 - \mathbb{P}[\hat{Y} = \hat{Y}'] && \text{(B.37)}\\
&= 1 - \mathbb{E}_{X,\hat{Y}}[\mathrm{p}(\hat{Y}' = \hat{Y} \mid X)] && \text{(B.38)}
\end{aligned}$$

$$= 1 - \mathbb{E}_{X,\hat{Y}}[\mathrm{p}(\hat{Y} \mid X)] \tag{B.39}$$
$$= 1 - \mathbb{E}[\mathrm{PredAcc}]. \tag{B.40}$$

The step from (B.37) to (B.38) is valid because $\hat{Y} \perp\!\!\!\perp \hat{Y}' \mid X$, and the step from (B.38) to (B.39) is valid because $\mathrm{p}(\hat{y}' \mid x) = \mathrm{p}(\hat{y} \mid x)$. Thus, we can rewrite Theorem B.1 as:

Lemma B.3. *The model* $\mathrm{p}(\hat{y} \mid x)$ *satisfies GDE, when*

$$\mathbb{E}[\mathrm{Acc}] = \mathbb{E}[\mathrm{p}(Y = \hat{Y} \mid X)] = \mathbb{E}[\mathrm{p}(\hat{Y} \mid X)] = \mathbb{E}[\mathrm{PredAcc}], \tag{B.41}$$

i.e. in expectation, the accuracy of the model equals the predicted accuracy of the model, or equivalently, the error of the model equals its predicted error.

Crucially, while Jiang et al. [2022] calls $1-\mathbb{E}_{X,\hat{Y}}[\mathrm{p}(\hat{Y}|X)]$ the (expected) disagreement rate $\mathbb{E}[\mathrm{Dis}]$, it actually is just the predicted error of the (Bayesian) model as a whole.

Equally important, all dependencies on Ω have vanished. Indeed, we will not use Ω anymore for the remainder of this section. This reproduces the corresponding remark from Jiang et al. [2022][9]:

Insight B.1. The theoretical statements in Jiang et al. [2022] can be made about any discriminative model with predictions $\mathrm{p}(y \mid x)$.

When is $\mathbb{E}_{X,\hat{Y}}[\mathrm{p}(Y = \hat{Y} \mid X)] = \mathbb{E}_{X,\hat{Y}}[\mathrm{p}(\hat{Y} \mid X)]$? Or in other words: when does $\mathrm{p}(Y = \hat{y} \mid x)$ equal $\mathrm{p}(\hat{Y} = \hat{y} \mid x)$ in expectation over $\mathrm{p}(x, y, \hat{y})$?

As a trivial sufficient condition, when the predictive distribution matches our data distribution—*i.e. when the model* $\mathrm{p}(\hat{y} \mid x)$ *is perfectly calibrated on average for all classes—and not only for the top-1 predicted class.* $ECE = 0$ is not sufficient because the standard calibration error only ensures that the data distribution and predictive distribution match for the top-1 predicted class [Nixon et al., 2019]. But class-wise calibration entails this equality.

Class-Wise and Class-Aggregated Calibration. To see this, we rewrite class-wise and class-aggregated calibration slightly by employing the following tautology:

$$\mathrm{p}(\hat{Y} = k \mid \mathrm{p}(\hat{Y} = k \mid X) = q) = q, \tag{B.42}$$

which is obviously true due its self-referential nature. We provide a formal proof in §B.2.7.4 in the appendix. Then we have the following equivalent definition:

Lemma B.4. *The model* $\mathrm{p}(\hat{y} \mid x)$ *satisfies* class-wise calibration *when for any* $q \in [0, 1]$ *and any class* $k \in [\mathtt{C}]$*:*

$$\mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) = \mathrm{p}(\hat{Y} = k, \mathrm{p}(\hat{Y} = k \mid X) = q). \tag{B.43}$$

Similarly, the model $\mathrm{p}(\hat{y}|x)$ *satisfies* class-aggregated calibration *when for any* $q \in [0, 1]$*:*

$$\mathrm{p}(\mathrm{p}(\hat{Y} = Y \mid X) = q) = \mathrm{p}(\mathrm{p}(\hat{Y} \mid X) = q), \tag{B.44}$$

and class-wise *calibration implies* class-aggregate *calibration.*

[9] The remark did not exist in the first preprint version.

The straightforward proof is found in §B.2.7.4 in the appendix.

Jiang et al. [2022] mention 'level sets' as intuition in their proof sketch. Here, we have been able to make this even clearer: class-aggregated calibration means that level-sets for accuracy $\mathrm{p}(\hat{Y} = Y \,|\, X)$ and predicted accuracy $\mathrm{p}(\hat{Y} \,|\, X)$—as random variables of Y and X, and $\hat{Y}$ and X, respectively—have equal measure, that is probability, for all q.

GDE. Now, class-aggregated calibration immediately and trivially implies GDE. To see this, we use the following property of expectations:

Lemma B.5. *For a random variable X, a function $t(x)$, and the random variable $T = t(X)$, it holds that*

$$\mathbb{E}_T[T] = \mathbb{E}[T] = \mathbb{E}_X[t(X)]. \tag{B.45}$$

This basic property states that we can either compute an expectation over T by integrating over $\mathrm{p}(T = t)$ or by integrating $t(x)$ over $\mathrm{p}(X = x)$. This is just a change of variable (push-forward of a measure).

We can use this property together with the class-aggregated calibration to see:

$$\begin{array}{ccc} \mathbb{E}[\mathrm{Acc}] & & \mathbb{E}[\mathrm{PredAcc}] \\ \| & & \| \\ \mathbb{E}_{X,Y}[\mathrm{p}(\hat{Y} = Y \mid X)] & & \mathbb{E}_{X,\hat{Y}}[\mathrm{p}(\hat{Y} \mid X)] \\ \| & & \| \\ \mathbb{E}[\mathrm{p}(\hat{Y} = Y \mid X)] & & \mathbb{E}[\mathrm{p}(\hat{Y} \mid X)] \\ \| & & \| \\ \mathbb{E}_{q \sim \mathrm{p}(\hat{Y} = Y \mid X)}[q] & = & \mathbb{E}_{q \sim \mathrm{p}(\hat{Y} \mid X)}[q] \quad , \end{array} \tag{B.46}$$

which is exactly Lemma B.3, where we start with the equality following from class-aggregated calibration and then apply Lemma B.5 along each side. Thus, GDE is but an expectation over class-aggregated calibration; we have:

Theorem B.6. *When a model $\mathrm{p}(\hat{y} \mid x)$ satisfies class-wise or class-aggregated calibration, it satisfies GDE.*

Proof. We can formalize the proof to be even more explicit and introduce two auxiliary random variables:

$$S \triangleq \mathrm{p}(\hat{Y} = Y \mid X), \tag{B.47}$$

as a transformed random variable of Y and X, and

$$T \triangleq \mathrm{p}(\hat{Y} \mid X), \tag{B.48}$$

as a transformed random variable of $\hat{Y}$ and X. Class-wise calibration implies class-aggregated calibration. Class-aggregated calibration then is $\mathrm{p}(S = q) = \mathrm{p}(T = q)$ (*). Writing out Equation B.46, we have

$$\mathbb{E}[\mathrm{p}(\hat{Y} = Y \mid X)] = \mathbb{E}_{X,Y}[S] = \mathbb{E}[S] = \mathbb{E}_S[S] \tag{B.49}$$

$$= \int \mathrm{p}(S = q)\, q \, dq \tag{B.50}$$

$$\overset{(*)}{=} \int \mathrm{p}(T = q)\, q\, dq \tag{B.51}$$

$$= \mathbb{E}_T[T] = \mathbb{E}[T] = \mathbb{E}_{X,\hat{Y}}[T] = \mathbb{E}[\mathrm{p}(\hat{Y} \mid X)], \tag{B.52}$$

which concludes the proof. □

The reader is invited to compare this derivation to the corresponding longer proof in the appendix of Jiang et al. [2022]. The fully probabilistic perspective greatly simplifies the results, and the proofs are straightforward.

CACE. Showing that CACE bounds the gap between test error and disagreement is also straightforward:

Theorem B.7. *For any model* $\mathrm{p}(\hat{y} \mid x)$*, we have:*

$$|\mathbb{E}[\text{TestError}] - \mathbb{E}[\text{Dis}]| \leq \text{CACE}.$$

Proof. First, we note that

$$\text{CACE} = \int_{q\in[0,1]} \Big|\, \mathrm{p}(\mathrm{p}(\hat{Y} = Y \mid X) = q) - \mathrm{p}(\mathrm{p}(\hat{Y} \mid X) = q)\Big| dq, \tag{B.53}$$

following the equivalences in the proof of Lemma B.4. Then using the triangle inequality for integrals and $0 \leq q \leq 1$, we obtain:

$$\begin{aligned}
&\text{CACE} && \text{(B.54)}\\
&= \int_{q\in[0,1]} \Big|\, \mathrm{p}(\mathrm{p}(\hat{Y} = Y \mid X) = q) - \mathrm{p}(\mathrm{p}(\hat{Y} = \hat{Y} \mid X) = q)\Big|\, dq && \text{(B.55)}\\
&\geq \int_{q\in[0,1]} q \Big|\, \mathrm{p}(\mathrm{p}(\hat{Y} = Y \mid X) = q) - \mathrm{p}(\mathrm{p}(\hat{Y} = \hat{Y} \mid X) = q)\Big|\, dq && \text{(B.56)}\\
&\geq \Big|\int_{q\in[0,1]} q\ \mathrm{p}(\mathrm{p}(\hat{Y} = Y \mid X) = q)\, dq - \int_{q\in[0,1]} q\ \mathrm{p}(\mathrm{p}(\hat{Y} \mid X) = q)\, dq\Big|. && \text{(B.57)}\\
&= \Big|\mathbb{E}[S] - \mathbb{E}[T]\Big| && \text{(B.58)}\\
&= \Big|\mathbb{E}[\text{TestError}] - \mathbb{E}[\text{Dis}]\Big|, && \text{(B.59)}
\end{aligned}$$

where we have used the monotonicity of integration in (B.56) and the triangle inequality in (B.57). □

The bound also serves as another—even simpler—proof for Theorem B.6:

Insight B.2. When the Bayesian model satisfies class-wise or class-aggregated calibration, we have CACE = 0 and thus $\mathbb{E}[\text{TestError}] = \mathbb{E}[\text{Dis}]$, i.e. the model satisfies GDE.

Furthermore, note again that a Bayesian model was not necessary for the last two theorems. The model parameters Ω were not mentioned—except for the specific definitions of TestError and Dis which depend on Ω following Jiang et al. [2022] but which we only use in expectation.

Moreover, we see that we can easily upper-bound CACE using the triangle inequality by 2, narrowing the statement in Jiang et al. [2022] that CACE can lie anywhere in $[0, \mathtt{C}]$:

Insight B.3. CACE ≤ 2.

Additionally, for completeness, we can also define the class-wise calibration error formally and show that it is bounded by CACE using the triangle inequality:

Definition B.5. The *class-wise calibration error (CWCE)* is defined as:

$$\text{CWCE} \triangleq \sum_k \int_{q\in[0,1]} \Big| \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) - \mathrm{p}(\hat{Y} = k, \mathrm{p}(\hat{Y} = k \mid X) = q) \Big|. \tag{B.60}$$

Lemma B.8. CWCE $\geq$ CACE $\geq |\,\mathbb{E}[\text{Acc}] - \mathbb{E}[\text{PredAcc}]|$.

Note that when we compute CACE empirically, we divide the dataset into several bins for different intervals of $\mathrm{p}(\hat{Y} = k \mid X)$. Jiang et al. [2022] use 15 bins. If we were to use a single bin, we would compute $|\,\mathbb{E}[\text{Acc}] - \mathbb{E}[\text{PredAcc}]|$ directly.

In §B.2.7.2 we show that CWCE has previously been introduced as 'adaptive calibration error' in Nixon et al. [2019] and CACE as 'static calibration error' (with noteworthy differences between Nixon et al. [2019] and its implementation).

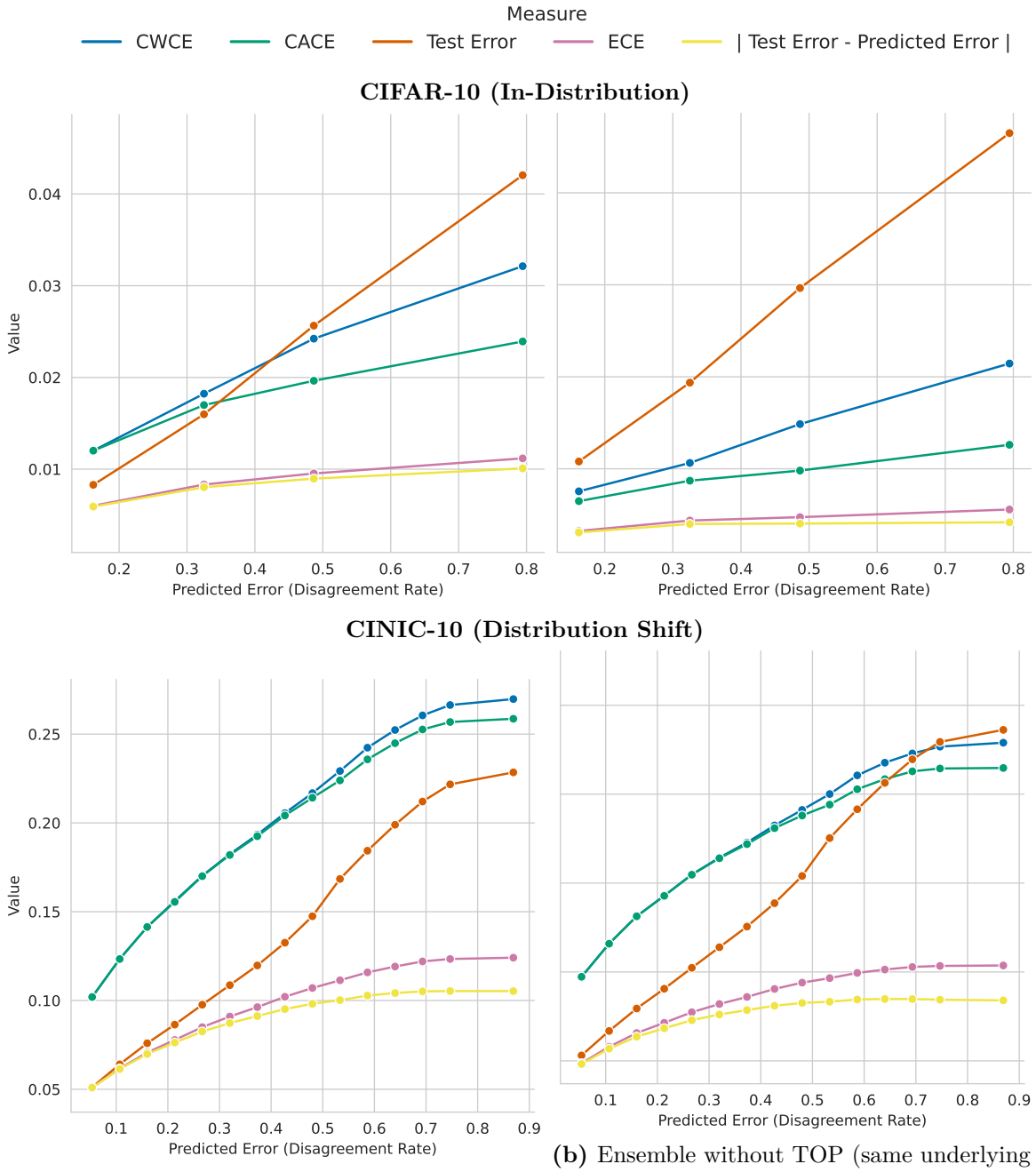
B.2.4 Deterioration of Calibration under Increasing Disagreement

Generally, we can only hope to trust model calibration for in-distribution data, while under distribution shift, the calibration ought to deteriorate. In our empirical falsification using models trained on CIFAR-10 and evaluated on the test sets of CIFAR-10 and CINIC-10, as a dataset with a distribution shift, we find in both cases that calibration deteriorates under increasing disagreement. We further examine ImageNET and PACS in §B.2.7.5. Most importantly though, calibration markedly worsens under distribution shift.

Specifically, we examine an ensemble of 25 WideResNet models [Zagoruyko and Komodakis, 2016] trained on CIFAR-10 [Krizhevsky, 2009] and evaluated on CIFAR-10 and CINIC-10 test data. CINIC-10 [Darlow et al., 2018] consists of CIFAR-10 and downscaled ImageNet samples for the same classes, and thus includes a distribution shift. The training setup follows the one described in Mukhoti et al. [2023], see appendix §B.2.7.5.

Figure B.5 shows rejection plots under increasing disagreement for in-distribution data (CIFAR-10) and under distribution shift (CINIC-10). The rejection plots threshold the dataset on increasing levels of the predicted error (disagreement rate)—which is a measure of epistemic uncertainty when there is no expected aleatoric uncertainty in the dataset. We examine ECE, class-aggregated calibration error (CACE), class-wise calibration error (CWCE), error $\mathbb{E}[\text{TestError}]$, and 'GDE gap', $|\,\mathbb{E}[\text{Acc}] - \mathbb{E}[\text{PredAcc}]|$, as the predicted error (disagreement rate), $\mathbb{E}[\text{Dis}] = 1 - \mathbb{E}[\text{PredAcc}]$, increases. We observe that all calibration metrics, ECE, CACE and CWCE, deteriorate under increasing disagreement, both in distribution and under distribution shift, and also worsen under distribution shift overall.

We also observe the same for ImageNet [Deng et al., 2009] and PACS [Li et al., 2017], which we show in appendix §B.2.7.5.



(a) Ensemble with TOP

(b) Ensemble without TOP (same underlying models)

Figure B.5: *Rejection Plot of Calibration Metrics for Increasing Disagreement In-Distribution (CIFAR-10) and Under Distribution Shift (CINIC-10).* Different calibration metrics (*ECE*, CWCE, CACE) vary across CIFAR-10 and CINIC-10 on an ensemble of 25 Wide-ResNet-28-10 model trained on CIFAR-10, depending on the rejection threshold of the predicted error (disagreement rate). Thus, calibration cannot be assumed constant for in-distribution data or under distribution shift. The test error increases almost linearly with the predicted error (disagreement rate), leading to 'GDE gap' |Test Error – Predicted Error| becoming almost flat, providing evidence for the empirical observations in Nakkiran and Bansal [2020]; Jiang et al. [2022]. The mean predicted error (disagreement rate) is shown on the x-axis. **(a)** shows results for an ensemble using TOP (following Jiang et al. [2022]), and **(b)** for a regular deep ensemble without TOP. The regular deep ensemble is better calibrated but has higher test error overall and lower test error for samples with small predicted error.

This is consistent with the experimental results of Ovadia et al. [2019] which examines dataset shifts. However, given that the calibration metrics change with the quantity of interest, we conclude that:

Insight B.4. The bound from Theorem B.2 might not have as much expressive power as hoped since the calibration metrics themselves deteriorate as the model becomes more 'uncertain' about the data.

At the same time, the 'GDE gap', which is the actual gap between test error and predicted error, flattens, and the test error develops an almost linear relationship with the predicted error (up to a bias). This shows that there seem to be intriguing empirical properties of deep ensemble as observed previously [Nakkiran and Bansal, 2020; Jiang et al., 2022]. However, they are not explained by the proposed calibration metrics[10].

As described previously, the results are not limited to Bayesian or version-space models but also apply to any model $\mathrm{p}(\hat{y} \mid x)$, including regular deep ensembles without TOP. In our experiment, we find that a regular deep ensemble is better calibrated than the same ensemble made to satisfy TOP. We hypothesize that each ensemble member's own predictive distribution is better calibrated than its one-hot outputs, yielding a better calibrated ensemble overall.

Given that all these calibration metrics require access to the labels, and we cannot assume the model to be calibrated under distribution shift, we might just as well use the labels directly to assess the test error.

B.2.5 Related Work

Here, we discuss connections to Bayesian model disagreement and epistemic uncertainty, as well as connections to information theory. We expand on these points in much greater detail in appendix §B.2.7.3.

Bayesian Model Disagreement. From a Bayesian perspective, as the epistemic uncertainty increases, we expect the model to become less reliable in its predictions. The predicted error of the model is a measure of the model's overall uncertainty, which is the total of aleatoric and epistemic uncertainty and thus correlated with epistemic uncertainty. Thus, we can hypothesize that as the predicted error increases, the model should become less reliable, which will be reflected in increasing calibration metrics. This is exactly what we have empirically validated in the previous section.

Connection to Information Theory. At first sight, Jiang et al. [2022] seems disconnected from information theory. However, we can draw a connection by using $\hat{\mathrm{H}}(p) \triangleq 1-p$ as a linear approximation for Shannon's information content $\mathrm{H}(p) = -\log p$. Semantically, both this approximation and Shannon's information content quantify surprise (i.e., prediction error). Both are 0 for certain events. For unlikely events, the former tends to 1 while the latter tends to $+\infty$.

This leads to common-sense definitions and statements from an information-theoretic point of view. We can even formulate parallel statements using information theory and see that the statements relate to total uncertainty and not epistemic uncertainty in a Bayesian sense.

[10] The simplest explanation is that very few samples have high predicted error and thus the rejection plots flatten. This is not true. For CINIC-10, the first bucket contains 50k samples, and each latter buckets adds additional ∼10k samples.

Other Related Literature. Beyond Jiang et al. [2022], this note offers a perspective on Granese et al. [2021] and Garg et al. [2022], which are proposing related approaches.

Granese et al. [2021] use the predicted error (disagreement rate), referred to as D_α, and the predicted top-1 error, D_β, to estimate when the model will be wrong. As noted in §B.2.7.3, the predicted error can be seen as an approximation of Shannon's entropy. Thus, D_α is effectively using an approximation of the prediction entropy for OoD detection and rejection classification. Similarly, D_β is the maximum class confidence. Both are well-known baselines for OoD detection [Hendrycks and Gimpel, 2017]. There is no ablation to see how D_α and D_β differ from these baselines. We leave this to future work. The paper frames the question of whether a model's predictions will be correct as binary classification problem on top of the underlying model's output probabilities and investigate this from a theoretical point of view. They also examine using input perturbations similar to Liang et al. [2018] and Lee et al. [2018b].

Garg et al. [2022] threshold the predictive entropy or maximum class confidence to estimate the test error under distribution shift. They estimate the threshold by calibrating it on in-distribution labeled data: the threshold is chosen such that the percentage of rejected in-distribution validation data approximately equals the test error on this in-distribution data. They call this approach *Average Thresholded Confidence (ATC)*. They find that ATC using entropy performs better than ATC using the maximum confidence and other approaches, including GDE. Their results show that ATC also degrades under increasing distribution shifts similar to what we have seen for GDE in §B.2.4 as the choice of threshold is explicitly tied to the in-distribution[11]. Garg et al. [2022] explicitly examine the theoretical limits when no further assumptions are made.

B.2.6 Discussion

We have found that the theoretical statements in Jiang et al. [2022] can be expressed and proven more concisely when using probabilistic notation for (Bayesian) models that output softmax probabilities.

Moreover, we empirically found the proposed calibration metrics to deteriorate under increasing disagreement for in-distribution data, and as expected, we have found the same behavior under distribution shifts.

While Jiang et al. [2022] are careful to qualify their results for distribution shifts, above results should give us pause: strong assumptions are still needed to conjecture about model generalization, and we need to beware of circular arguments.

B.2.7 Details

B.2.7.1 Equivalent Definitions

Jiang et al. [2022] defines a *hypothesis space* $\mathcal{H}$. In the literature, this is also sometimes called a version space. The hypothesis space induced by a stochastic training algorithm $\mathcal{A}$ is named $\mathcal{H}_\mathcal{A}$.

We can identify each hypothesis $h : \mathcal{X} \to [\mathtt{C}]$ with itself as parameter $\omega_h = h$ and define $\mathrm{p}(\omega_h)$ as a uniform distribution over all parameters/hypotheses in $\mathcal{H}_\mathcal{A}$. This has the advantage of formalizing the distribution from which hypothesis are drawn

[11] See also Figures 7, 8, and 9 in the appendix of Garg et al. [2022]

$(h \sim \mathcal{H}_{\mathcal{A}})$, which is not made explicit in Jiang et al. [2022]. $h(x) = k$ then becomes $\arg\max_{\hat{y}} \mathrm{p}(\hat{y} \mid x, \omega_h) = k$. Moreover, as $\mathrm{p}(\hat{y}, \omega \mid x)$ satisfies TOP, we have[12]

$$\text{“}\mathbb{1}\{h(x) = k\}\text{”} = \mathrm{p}(\hat{Y} = k \mid x, \omega_h). \tag{B.61}$$

Thus, “$\mathrm{TestErr}_{\mathscr{D}}(h) \triangleq \mathbb{E}_{\mathscr{D}}[\mathbb{1}[h(X) \neq Y]]$” is equivalent to:

$$\begin{aligned}
\text{“}\,\mathrm{TestErr}_{\mathscr{D}}(h) &= \mathbb{E}_{\mathscr{D}}[\mathbb{1}[h(X) \neq Y]]\text{”} && \text{(B.62)}\\
&= \mathbb{E}_{X,Y}[\mathrm{p}(\hat{Y} \neq Y \mid X, \omega_h)] && \text{(B.63)}\\
&= \mathbb{E}_{X}[\mathbb{P}[\hat{Y} \neq Y \mid X, \omega_h]] && \text{(B.64)}\\
&= \mathbb{P}[\hat{Y} \neq Y \mid \omega_h] && \text{(B.65)}\\
&= \mathrm{TestError}(\omega_h). && \text{(B.66)}
\end{aligned}$$

Similarly, “$\mathrm{Dis}_{\mathscr{D}}(h, h') \triangleq \mathbb{E}_{\mathscr{D}}[\mathbb{1}[h(X) \neq h'(X)]]$” is equivalent to:

$$\begin{aligned}
\text{“}\,\mathrm{Dis}_{\mathscr{D}}(h, h') \triangleq \mathbb{E}_{\mathscr{D}}[\mathbb{1}[h(X) \neq h'(X)]]\text{”} && \text{(B.67)}\\
= \mathbb{E}_{\hat{Y},\hat{Y}'}[\mathbb{P}[\hat{Y} \neq \hat{Y}' \mid \omega_h, \omega_{h'}]] && \text{(B.68)}\\
= \mathrm{Dis}(\omega_h, \omega_{h'}). && \text{(B.69)}
\end{aligned}$$

Further, “$\tilde{h}_k(x) \triangleq \mathbb{E}_{\mathscr{H}_{\mathcal{A}}}[\mathbb{1}[h(x) = k]]$” is equivalent to:

$$\begin{aligned}
\text{“}\tilde{h}_k(x) \triangleq \mathbb{E}_{\mathscr{H}_{\mathcal{A}}}[\mathbb{1}[h(x) = k]]\text{”} && \text{(B.70)}\\
= \mathbb{E}_{\Omega}[\mathrm{p}(\hat{Y} = k \mid x, \Omega)] && \text{(B.71)}\\
= \mathrm{p}(\hat{Y} = k \mid x). && \text{(B.72)}
\end{aligned}$$

For the GDE, “$\mathbb{E}_{h,h' \sim \mathscr{H}_{\mathcal{A}}}[\mathrm{Dis}_{\mathscr{D}}(h, h')] = \mathbb{E}_{h \sim \mathscr{H}_{\mathcal{A}}}[\mathrm{TestErr}(h)]$” is equivalent to:

$$\begin{aligned}
&\text{“}\mathbb{E}_{h,h' \sim \mathscr{H}_{\mathcal{A}}}[\mathrm{Dis}_{\mathscr{D}}(h, h')] = \mathbb{E}_{h \sim \mathscr{H}_{\mathcal{A}}}[\mathrm{TestErr}(h)]\text{”}\\
&\Leftrightarrow \mathbb{E}_{\Omega,\Omega'}[\mathrm{Dis}(\Omega, \Omega')] = \mathbb{E}_{\Omega}[\mathrm{TestError}(\Omega)]. && \text{(B.73)}
\end{aligned}$$

For the class-wise calibration, “$p(Y = k \mid \tilde{h}_k(X) = q) = q$” is equivalent to:

$$\begin{aligned}
\text{“}p(Y = k \mid \tilde{h}_k(X) = q) = q\text{”} && \text{(B.74)}\\
\Leftrightarrow \mathrm{p}(Y = k \mid \mathrm{p}(\hat{Y} = k \mid X) = q) = q. && \text{(B.75)}
\end{aligned}$$

For the class-aggregated calibration, “$\frac{\sum_{k=0}^{K-1} p(Y=k, \tilde{h}_k(X)=q)}{\sum_{k=0}^{K-1} p(\tilde{h}_k(X)=q)} = q$” (and note in Jiang et al. [2022], class indices run from $0..K-1$) is equivalent to:

$$\begin{aligned}
\text{“}\frac{\sum_{k=0}^{K-1} p(Y = k, \tilde{h}_k(X) = q)}{\sum_{k=0}^{K-1} p(\tilde{h}_k(X) = q)} = q\text{”} && \text{(B.76)}\\
\Leftrightarrow \frac{\sum_{k=1}^{\mathsf{C}} \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q)}{\sum_{k=1}^{\mathsf{C}} \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q)} = q && \text{(B.77)}
\end{aligned}$$

[12] We put definitions and expressions written using the notation and variables from Jiang et al. [2022] inside quotation marks “” to avoid ambiguities.

$$\Leftrightarrow \sum_{k=1}^{\mathsf{C}} \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q)$$
$$= q \sum_{k=1}^{\mathsf{C}} \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q). \tag{B.78}$$

Finally, for the class-aggregated calibration error, the definition is equivalent to:

“ $\mathrm{CACE}_{\mathscr{D}}(\tilde{h})$

$$\triangleq \int_{q\in[0,1]} \left| \frac{\sum_k p\left(Y = k, \tilde{h}_k(X) = q\right)}{\sum_k p\left(\tilde{h}_k(X) = q\right)} - q \right| \cdot \sum_k p\left(\tilde{h}_k(X) = q\right) dq \tag{B.79}$$

$$= \int_{q\in[0,1]} \left| \sum_k p\left(Y = k, \tilde{h}_k(X) = q\right) - q \sum_k p\left(\tilde{h}_k(X) = q\right) \right| dq'' \tag{B.80}$$

$$= \int_{q\in[0,1]} \left| \sum_k \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) - q \sum_k \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q) \right| dq \tag{B.81}$$

B.2.7.2 Comparison of CACE and CWCE with calibration metrics with ‘adaptive calibration error’ and ‘static calibration error’

Nixon et al. [2019] examine shortcomings of the ECE metric and identify a lack of class conditionality, adaptivity and the focus on the maximum probability (argmax class) as issues. They suggest an adaptive calibration error which uses adaptive binning and averages of the calibration error separately for each class, thus equivalent to the class-wise calibration error and class-wise calibration (up to adaptive vs. even binning). In the paper, the static calibration error is defined as ACE with even instead of adaptive binning. However, in the widely used implementation[13], SCE is defined as equivalent to the class-aggregated calibration error.

B.2.7.3 Expanded Discussion

Here, we discuss connections to Bayesian model disagreement and epistemic uncertainty, as well as connections to information theory, the bias-variance trade-off, and prior literature.

Bayesian Model Disagreement From a Bayesian perspective, as the epistemic uncertainty increases, we expect the model to become less reliable in its predictions. The predicted error of the model is a measure of the overall uncertainty of the model which is the total of aleatoric and epistemic uncertainty, and thus correlated with epistemic uncertainty. Thus, we can hypothesize that as the predicted error increases, the model should become less reliable, which will be reflected in increasing calibration metrics. This is what we have empirically validated in the previous section. We can expand on the connection to the Bayesian perspective. In particular, we can connect the statements of Jiang et al. [2022] to a well-known Bayesian measure of model disagreement.

In §B.2.7.5, we also report empirical results for rejection plots based on Bayesian model disagreement instead of predicted error.

[13] https://github.com/google-research/robustness_metrics/blob/baa47fbe38f80913590545fe7c199898f9aff349/robustness_metrics/metrics/uncertainty.py#L1585, added in April 2021

Connection to Information Theory At first sight, Jiang et al. [2022] seems disconnected from information theory. However, we can recover statements by using $\hat{\mathrm{H}}(p) \triangleq 1 - p$ as a linear approximation for Shannon's information content $\mathrm{H}(p)$:

$$\hat{\mathrm{H}}(p) = 1 - p \leq -\log p = \mathrm{H}(p). \tag{B.82}$$

$\hat{\mathrm{H}}(p)$ is just the first-order Taylor expansion of $\mathrm{H}(p) = -\log p$ around 1. Semantically, both Shannon's information content and this approximation quantify surprise. Both are 0 for certain events. For unlikely events, the former tends to $+\infty$ while the latter tends to 1.

We can define an *approximate entropy* $\hat{\mathrm{H}}[X]$ using H':

$$\hat{\mathrm{H}}[X] \triangleq \mathbb{E}[\hat{\mathrm{H}}(\mathrm{p}(X))] = 1 - \mathbb{E}[\mathrm{p}(x)] = 1 - \sum_x \mathrm{p}(x)^2, \tag{B.83}$$

and an *approximate mutual information* $\hat{\mathrm{I}}[X;Y]$:

$$\hat{\mathrm{I}}[X;Y] \triangleq \hat{\mathrm{H}}[X] - \hat{\mathrm{H}}[X \mid Y] = \hat{\mathrm{H}}[X] - \mathbb{E}_{\mathrm{p}(y)} \hat{\mathrm{H}}[X \mid y], \tag{B.84}$$

following the semantic notion of mutual information as expected information gain in §B.2.1.

$\hat{\mathrm{I}}[\hat{Y};\Omega \mid x]$ as Covariance Trace. This approximate mutual information has a surprisingly nice property, which was detailed in Smith and Gal [2018] originally:

Proposition B.9. *The approximate mutual information $\hat{\mathrm{I}}[\hat{Y};\Omega \mid x]$ is equal the sum of the variances of $\hat{y} \mid x, \Omega$ over all $\hat{y}$:*

$$\hat{\mathrm{I}}[\hat{Y};\Omega \mid x] = \sum_{\hat{y}=1}^{K} \mathrm{Var}_\Omega[\mathrm{p}(\hat{y} \mid x, \Omega)] \geq 0. \tag{B.85}$$

We present a proof in §B.2.7.4. The sum of variances of the predictive probabilities (or trace of the respective covariance matrix) is a common proxy for epistemic uncertainty [Gal et al., 2017], and here the mutual information $\hat{\mathrm{I}}[\hat{Y};\Omega \mid x]$ using $\hat{\mathrm{H}}$ is just that. This gives evidence that these definitions are sensible and connects them to other prior Bayesian literature. Importantly, this also shows that $\hat{\mathrm{H}}[\hat{Y} \mid x] \geq \hat{\mathrm{H}}[\hat{Y} \mid x, \Omega]$.

Connection to Jiang et al. [2022]. As random variable of X and Y, $\hat{\mathrm{H}}[\hat{Y} = Y \mid X]$ is the test error:

$$\hat{\mathrm{H}}[\hat{Y} = Y \mid X] = 1 - \mathrm{p}(\hat{Y} = Y \mid X) = \mathrm{TestError}. \tag{B.86}$$

Thus, the approximate cross-entropy

$$\hat{\mathrm{H}}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) = \mathbb{E}_{\mathrm{p}(Y|X)}[\hat{\mathrm{H}}(\mathrm{p}(\hat{Y} = Y \mid X))] \tag{B.87}$$

is the expected test error $\mathbb{E}[\mathrm{TestError}]$.

Similarly, when TOP is fulfilled, the mutual information $\hat{\mathrm{I}}[\hat{Y};\Omega \mid X]$ is the expected disagreement rate $\mathbb{E}[\mathrm{Dis}]$. That is, when $\hat{Y} \mid X, \Omega$ is one-hot, we have:

$$\hat{\mathrm{H}}[\hat{Y} \mid X, \Omega] = 1 - \mathbb{E}_X \underbrace{\mathbb{E}_{\hat{Y}}[\mathrm{p}(\hat{Y} \mid X, \Omega) \mid X]}_{=1} = 0, \tag{B.88}$$

and thus:

$$\hat{\mathrm{I}}[\hat{Y}; \Omega \mid X] = \hat{\mathrm{H}}[\hat{Y} \mid X] - \hat{\mathrm{H}}[\hat{Y} \mid X, \Omega] \tag{B.89}$$
$$= \hat{\mathrm{H}}[\hat{Y} \mid X] \tag{B.90}$$
$$= 1 - \mathbb{E}_{X,\hat{Y}}[\mathrm{p}(\hat{Y} \mid X)] \tag{B.91}$$
$$= \mathbb{E}[\mathrm{Dis}]. \tag{B.92}$$

Lemma B.10. *When the model* $\mathrm{p}(\hat{y} \mid x, \omega)$ *satisfies TOP, the GDE is equivalent to:*

$$\hat{\mathrm{H}}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) = \hat{\mathrm{I}}[\hat{Y}; \Omega \mid X]. \tag{B.93}$$

This relates the approximate cross-entropy loss (test error) to the approximate Bayesian model disagreement.

Without TOP. If TOP does not hold, the *actual* expected disagreement $\hat{\mathrm{I}}[\hat{Y}; \Omega \mid x]$ lower-bounds the "expected disagreement rate" $\mathbb{E}[\mathrm{Dis}]$, which then equals the expected *predicted* error $1 - \mathbb{E}_{X,\hat{Y}}[\mathrm{p}(\hat{Y} \mid X)]$ when we have GDE. We have the following general equivalence to GDE:

Lemma B.11. *For a model* $\mathrm{p}(\hat{y} \mid x)$*, the GDE is equivalent to:*

$$\hat{\mathrm{H}}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) = \hat{\mathrm{H}}[\hat{Y} \mid X] \geq \hat{\mathrm{I}}[\hat{Y}; \Omega \mid X]. \tag{B.94}$$

The other statements and proofs translate likewise, and intuitively seem sensible from an information-theoretic perspective. We can go further and directly establish analogous properties using information theory in the next subsection.

Information-Theoretic Version Here, we derive an information-theoretic version of the GDE both under the assumption of TOP and without. Importantly, we will not require a Bayesian model for any of the main statements as they hold for any model $\mathrm{p}(\hat{y} \mid x)$. We show that we can artificially introduce a connection to disagreement using TOP.

Information-Theoretic GDE. We have already introduced the BALD equation **??**, which connects expected disagreement and predictive uncertainty:

$$\mathrm{I}[\hat{Y}; \Omega \mid x] = \mathrm{H}[\hat{Y} \mid x] - \mathrm{H}[\hat{Y} \mid x, \Omega]$$

The expected disagreement is measured by the mutual information $\mathrm{I}[\hat{Y}; \Omega \mid X]$, and the prediction error is measured by the cross-entropy of the predictive distribution under the true data generating distribution $\mathrm{H}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X))$. Indeed, the test error is bounded by it [Kirsch et al., 2020]:

$$\mathrm{p}(Y \neq \hat{Y}) \leq 1 - e^{-\mathrm{H}(\mathrm{p}(Y|X) \| \mathrm{p}(\hat{Y} = Y|X))}. \tag{B.95}$$

When our model fulfills TOP, we have $\mathrm{H}[\hat{Y} \mid X, \Omega] = 0$, and thus $\mathrm{I}[\hat{Y}; \Omega \mid X] = \mathrm{H}[\hat{Y} \mid X]$. The expected disagreement then equals the predicted label uncertainty $\mathrm{H}[\hat{Y} \mid X]$. Generally, we can define an 'entropic GDE':

Definition B.6. A model $\mathrm{p}(\hat{y} \mid x)$ satisfies entropic GDE, when:

$$\mathrm{H}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) = \mathrm{H}[\hat{Y} \mid X]. \tag{B.96}$$

Lemma B.12. *When a Bayesian model* $\mathrm{p}(\hat{y}, \omega \mid x)$ *satisfies TOP, entropic GDE is equivalent to*

$$\mathrm{H}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) = \mathrm{I}[\hat{Y}; \Omega \mid X]. \tag{B.97}$$

The latter is close to GDE, especially when comparing to the previous section.

We can formulate an entropic class-aggregated calibration by connecting $\mathrm{H}[\hat{y} \,|\, x]$ with $\mathrm{H}[y \,|\, x]$. That is, instead of using probabilities, we use Shannon's information-content:

Definition B.7. The model $\mathrm{p}(\hat{y} \,|\, x)$ satisfies *entropic class-aggregated calibration* when for any $q \geq 0$:

$$\mathrm{p}(\mathrm{H}[\hat{Y} = Y \mid X] = q) = \mathrm{p}(\mathrm{H}[\hat{Y} = \hat{Y} \mid X] = q). \tag{B.98}$$

Similarly, we can define the *entropic class-aggregated calibration error (ECACE)*:

$$\begin{aligned} \mathrm{ECACE} \triangleq \int_{q \in [0,\infty)} \Big| & \mathrm{p}(\mathrm{H}[\hat{Y} = Y \mid X] = q) \\ & - \mathrm{p}(\mathrm{H}[\hat{Y} = \hat{Y} \mid X] = q) \Big| dq. \end{aligned} \tag{B.99}$$

As $-\log p$ is strictly monotonic and thus invertible for non-negative p, entropic class-aggregated calibration and class-aggregated calibration are equivalent. ECACE and CACE are not, though.

The expectation of the transformed random variable $\mathrm{H}[\hat{Y} = Y \mid X]$ (in Y and X) is just the cross-entropy:

$$\mathbb{E}_{X,Y}\, \mathrm{H}[\hat{Y} = Y \mid X] = \mathbb{E}_{\mathrm{p}(x,y)}\, \mathrm{H}[\hat{Y} = y \mid X] = \mathrm{H}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)). \tag{B.100}$$

Using this notation, and analogous to Theorem B.2, we can show:

Theorem B.13. *When* $\mathrm{H}[\hat{y} \mid x] = -\log \mathrm{p}(\hat{y} \mid x)$ *is upper-bounded by* L *for all* $\hat{y}$ *and* x*, we have:*

$$\mathrm{ECACE} \geq \frac{1}{L} \Big| \mathrm{H}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) - \mathrm{H}[\hat{Y} \mid X] \Big|, \tag{B.101}$$

and when the model satisfies TOP, equivalently:

$$= \frac{1}{L} \Big| \mathrm{H}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) - \mathrm{I}[\hat{Y}; \Omega \mid X] \Big|. \tag{B.102}$$

There might be better conditions than the upper-bound above, but this bound is in the spirit of Jiang et al. [2022]. Indeed, the proof of Theorem B.2 is the same, except that we use $q \leq L$ instead of $q \leq 1$. Finally, when the model satisfies entropic class-aggregated calibration, ECACE $= 0$, cross-entropy (or negative expected log likelihood) equals disagreement (respectively, predicted label uncertainty when TOP does not hold). Thus, we have:

Theorem B.14. *When the model* $\mathrm{p}(\hat{y} \,|\, x)$ *satisfies* entropic class-aggregated calibration*, it trivially also satisfies entropic GDE:*

$$\mathrm{H}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) = \mathrm{H}[\hat{Y} \mid X] \geq \mathrm{I}[\hat{Y}; \Omega \mid X], \tag{B.103}$$

and when TOP holds:

$$\mathrm{H}(\mathrm{p}(Y \mid X) \parallel \mathrm{p}(\hat{Y} = Y \mid X)) = \mathrm{I}[\hat{Y}; \Omega \mid X]. \tag{B.104}$$

Without TOP. Again, if we do not expect one-hot predictions for our ensemble members, the analogy put forward in Jiang et al. [2022] breaks down because the Bayesian disagreement $\mathrm{I}[\hat{Y}; \Omega \mid X]$ only lower bounds the predicted label uncertainty $\mathrm{H}[\hat{Y} \mid X]$ and can not be connected to ECACE the same way. But this also breaks down in the regular version in Jiang et al. [2022].

B.2.7.4 Additional Proofs

Lemma B.15. *For a model* $\mathrm{p}(\hat{y} \mid x)$*, we have for all* $k \in [K]$ *and* $q \in [0, 1]$*:*

$$\mathrm{p}(\hat{Y} = k \mid \mathrm{p}(\hat{Y} = k \mid X) = q) = q, \tag{B.105}$$

when the left-hand side is well-defined.

Proof. This is equivalent to

$$\mathrm{p}(\hat{Y} = k, \mathrm{p}(\hat{Y} = k \mid X) = q) = q \ \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q), \tag{B.106}$$

as the conditional probability is either defined or $\mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q) = 0$. Assume the former. Let $\mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q) > 0$. Introducing the auxiliary random variable $T_k \triangleq \mathrm{p}(\hat{Y} = k \mid X)$ as a transformed random variable of X, we have

$$\mathrm{p}(\hat{Y} = k, T_k = q) = q \ \mathrm{p}(T_k = q). \tag{B.107}$$

We can write the probability as an expectation over an indicator function

$$\begin{aligned}
&\mathrm{p}(\hat{Y} = k, T_k = q) && \text{(B.108)}\\
&\quad = \mathbb{E}_{X,\hat{Y}}[\mathbb{1}\{\hat{Y} = k, T_k(X) = q\}] && \text{(B.109)}\\
&\quad = \mathbb{E}_{X,\hat{Y}}[\mathbb{1}\{\hat{Y} = k\} \, \mathbb{1}\{T_k(X) = q\}] && \text{(B.110)}\\
&\quad = \mathbb{E}_X[\mathbb{1}\{T_k(X) = q\} \ \mathbb{E}_{\hat{Y}}[\mathbb{1}\{\hat{Y} = k\} \mid X]] && \text{(B.111)}\\
&\quad = \mathbb{E}_X[\mathbb{1}\{T_k(X) = q\} \ \mathrm{p}(\hat{Y} = k \mid X)]. && \text{(B.112)}
\end{aligned}$$

Importantly, if $\mathbb{1}\{T_k(x) = q\} = 1$ for an x, we have $T_k(x) = \mathrm{p}(\hat{Y} = k \mid x) = q$, and otherwise, we multiply with 0. Thus, this is equivalent to

$$\begin{aligned}
&= \mathbb{E}_X[\mathbb{1}\{T_k(X) = q\} \, q] && \text{(B.113)}\\
&= q \, \mathbb{E}_X[\mathbb{1}\{T_k(X) = q\}] && \text{(B.114)}\\
&= q \, \mathrm{p}(T_k(X) = q). && \text{(B.115)}
\end{aligned}$$

□

Lemma B.4. *The model* $\mathrm{p}(\hat{y} \mid x)$ *satisfies* class-wise calibration *when for any* $q \in [0, 1]$ *and any class* $k \in [\mathtt{C}]$*:*

$$\mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) = \mathrm{p}(\hat{Y} = k, \mathrm{p}(\hat{Y} = k \mid X) = q). \tag{B.43}$$

Similarly, the model $\mathrm{p}(\hat{y} \mid x)$ *satisfies* class-aggregated calibration *when for any* $q \in [0, 1]$*:*

$$\mathrm{p}(\mathrm{p}(\hat{Y} = Y \mid X) = q) = \mathrm{p}(\mathrm{p}(\hat{Y} \mid X) = q), \tag{B.44}$$

and class-wise *calibration implies* class-aggregate *calibration.*

Proof. Beginning from

$$\mathrm{p}(Y = k \mid \mathrm{p}(\hat{Y} = k \mid X) = q) = q, \tag{B.116}$$

we expand the conditional probability to

$$\Leftrightarrow \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) = q\ \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q), \tag{B.117}$$

and substitute Equation B.42 into the outer q, obtaining the first equivalence

$$\Leftrightarrow \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) = \mathrm{p}(\hat{Y} = k, \mathrm{p}(\hat{Y} = k \mid X) = q). \tag{B.118}$$

For the second equivalence, we follow the same approach. Beginning from

$$\sum_k \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) = q \sum_k \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q), \tag{B.119}$$

we pull the outer q into the sum and expand using (B.42)

$$\Leftrightarrow \sum_k \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = k \mid X) = q) = \sum_k q\, \mathrm{p}(\mathrm{p}(\hat{Y} = k \mid X) = q) = \sum_k \mathrm{p}(\hat{Y} = k, \mathrm{p}(\hat{Y} = k \mid X) = q). \tag{B.120}$$

In the inner expression, k is tied to Y on the left-hand side and $\hat{Y}$ on the right-hand side, so we have

$$\Leftrightarrow \sum_k \mathrm{p}(Y = k, \mathrm{p}(\hat{Y} = Y \mid X) = q) = \sum_k \mathrm{p}(\hat{Y} = k, \mathrm{p}(\hat{Y} \mid X) = q). \tag{B.121}$$

Summing over k, marginalizes out $Y = k$ and $\hat{Y} = k$ respectively, yielding the second equivalence

$$\Leftrightarrow \mathrm{p}(\mathrm{p}(\hat{Y} = Y \mid X) = q) = \mathrm{p}(\mathrm{p}(\hat{Y} \mid X) = q). \tag{B.122}$$

Finally, class-wise calibration implies class-aggregated calibration as summing over different k in (B.118), which is equivalent to class-wise calibration, yields (B.120), which is equivalent to class-aggregated calibration. □

Proposition B.9. *The approximate mutual information* $\hat{\mathrm{I}}[\hat{Y}; \Omega \mid x]$ *is equal the sum of the variances of* $\hat{y} \mid x, \Omega$ *over all* $\hat{y}$*:*

$$\hat{\mathrm{I}}[\hat{Y}; \Omega \mid x] = \sum_{\hat{y}=1}^{K} \mathrm{Var}_\Omega[\mathrm{p}(\hat{y} \mid x, \Omega)] \geq 0. \tag{B.85}$$

Proof. We show that both sides are equal:

$$\begin{aligned}
\hat{\mathrm{I}}[\hat{Y}; \Omega \mid x] &= \hat{\mathrm{H}}[\hat{Y} \mid x] - \hat{\mathrm{H}}[\hat{Y} \mid x, \Omega] && \text{(B.123)}\\
&= \mathbb{E}_{\hat{Y}}[1 - \mathrm{p}(\hat{Y} \mid x)] - \mathbb{E}_{\hat{Y},\Omega}[1 - \mathrm{p}(\hat{Y} \mid x, \Omega)] && \text{(B.124)}\\
&= \mathbb{E}_{\hat{Y},\Omega}[\mathrm{p}(\hat{Y} \mid x, \Omega)] - \mathbb{E}_{\hat{Y}}[\mathrm{p}(\hat{Y} \mid x)] && \text{(B.125)}\\
&= \mathbb{E}_\Omega\, \mathbb{E}_{\mathrm{p}(\hat{y},x,\Omega)}[\mathrm{p}(\hat{y} \mid x, \Omega)] - \mathbb{E}_{\mathrm{p}(\hat{y}|x)}\, \mathrm{p}(\hat{y} \mid x) && \text{(B.126)}
\end{aligned}$$

$$= \mathbb{E}_\Omega \left[\sum_{\hat{y}=1}^{K} \mathrm{p}(\hat{y} \mid x, \Omega)^2 \right] - \sum_{\hat{y}=1}^{K} \mathbb{E}_\Omega [\mathrm{p}(\hat{y} \mid x, \Omega)]^2 \tag{B.127}$$

$$= \sum_{\hat{y}=1}^{K} \mathbb{E}_\Omega \left[\mathrm{p}(\hat{y} \mid x, \Omega)^2 \right] - \mathbb{E}_\Omega [\mathrm{p}(\hat{y} \mid x, \Omega)]^2 \tag{B.128}$$

$$= \sum_{\hat{y}=1}^{K} \mathrm{Var}_\Omega [\mathrm{p}(\hat{y} \mid x, \Omega)] \tag{B.129}$$

$$\geq 0, \tag{B.130}$$

where we have used that $\mathbb{E}_{\mathrm{p}(\hat{y}|x)}\, \mathrm{p}(\hat{y} \mid x) = \sum_{\hat{y}=1}^{K} \mathrm{p}(\hat{y} \mid x)^2$. □

B.2.7.5 Empirical Validation of Calibration Deterioration under Increasing Disagreement

Here, we discuss additional details to allow for reproduction and present results on additional datasets. In addition to the experiments on CIFAR-10 [Krizhevsky, 2009] and CINIC-10 [Darlow et al., 2018], we report results for ImageNet [Deng et al., 2009] (in-distribution) using an ensemble of pretrained models and PACS [Li et al., 2017] (distribution shift) where we fine-tune ImageNet models on PACS' 'photo' domain, which is close to ImageNet as source domain, and evaluate it on PACS' 'art painting', 'sketch', and 'cartoon' domains. We use all three domains together for distribution shift evaluation to have more samples for the rejection plots.

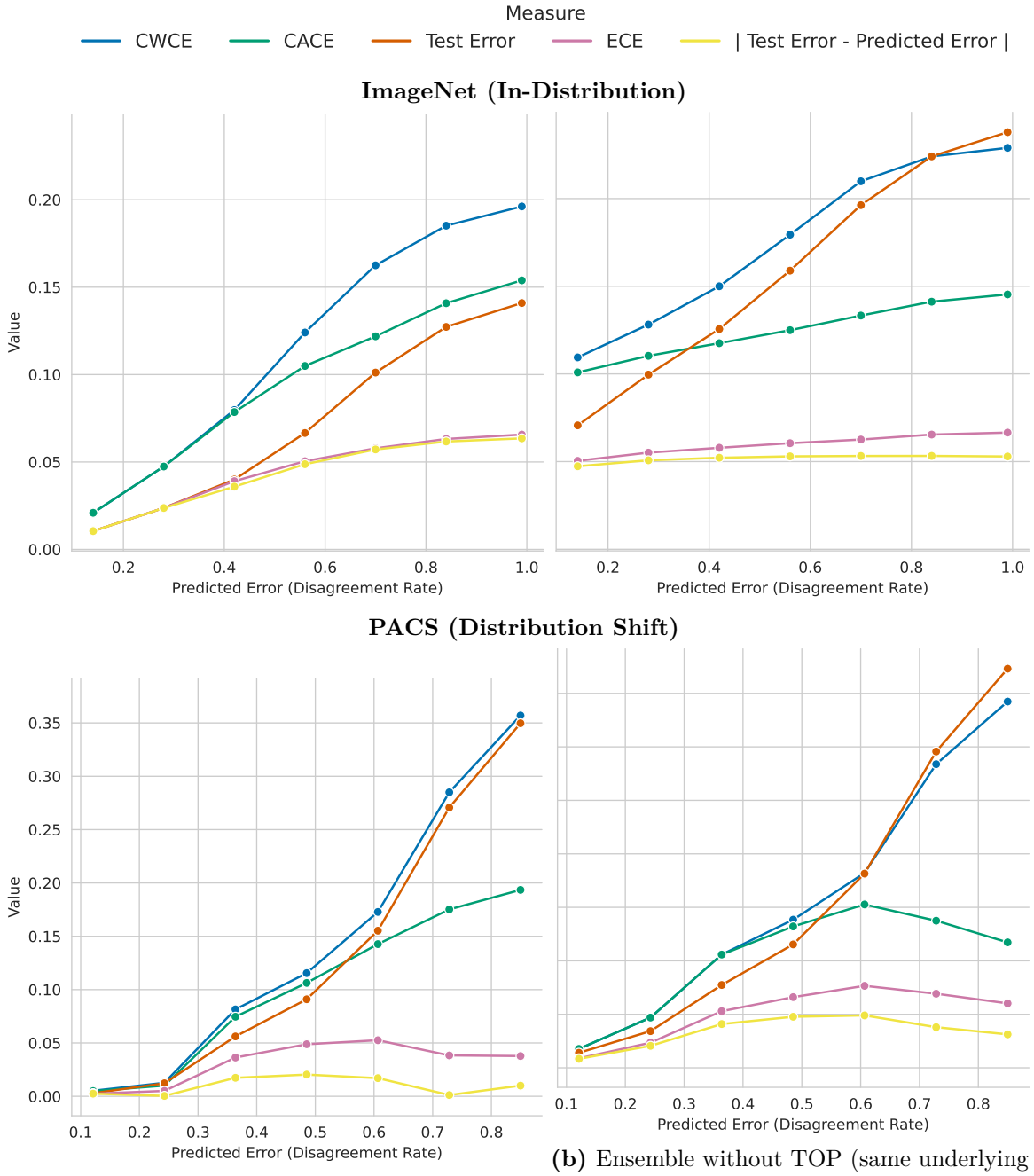
Experiment Setup We use PyTorch [Paszke et al., 2019] for all experiments.

CIFAR-10 and CINIC-10. We follow the training setup from Mukhoti et al. [2023]: we train 25 WideResNet-28-10 models [Zagoruyko and Komodakis, 2016] for 350 epochs on CIFAR-10. We use SGD with a learning rate of 0.1 and momentum of 0.9. We use a learning rate schedule with a decay of 10 at 150 and 250 epochs.

ImageNet and PACS. We use pretrained models with various architectures (specifically: ResNet-152-D [He et al., 2018], BEiT-L/16 [Bao et al., 2021], ConvNext-L [Liu et al., 2022], DeiT3-L/16 [Touvron et al., 2020], and ViT-B/16 [Dosovitskiy et al., 2020]) from the timm package [Wightman, 2019] as base models. We freeze all weights except for the final linear layer, which we fine-tune on PACS' 'photo' domain using Adam [Kingma and Ba, 2015] with learning rate 5×10^{-3} and batch size 128 for 1000 steps. We then build an ensemble using these different models.

Additional Results In Figure B.6, we see that for ImageNet and PACS, the calibration metrics behave like for CIFAR-10 and CINIC-10, matching the described behavior in the main text. We use 5 models from each of the enumerated architectures to build an ensemble of 25 models. Individual architectures also behave as expected as we ablate in Figure B.9.

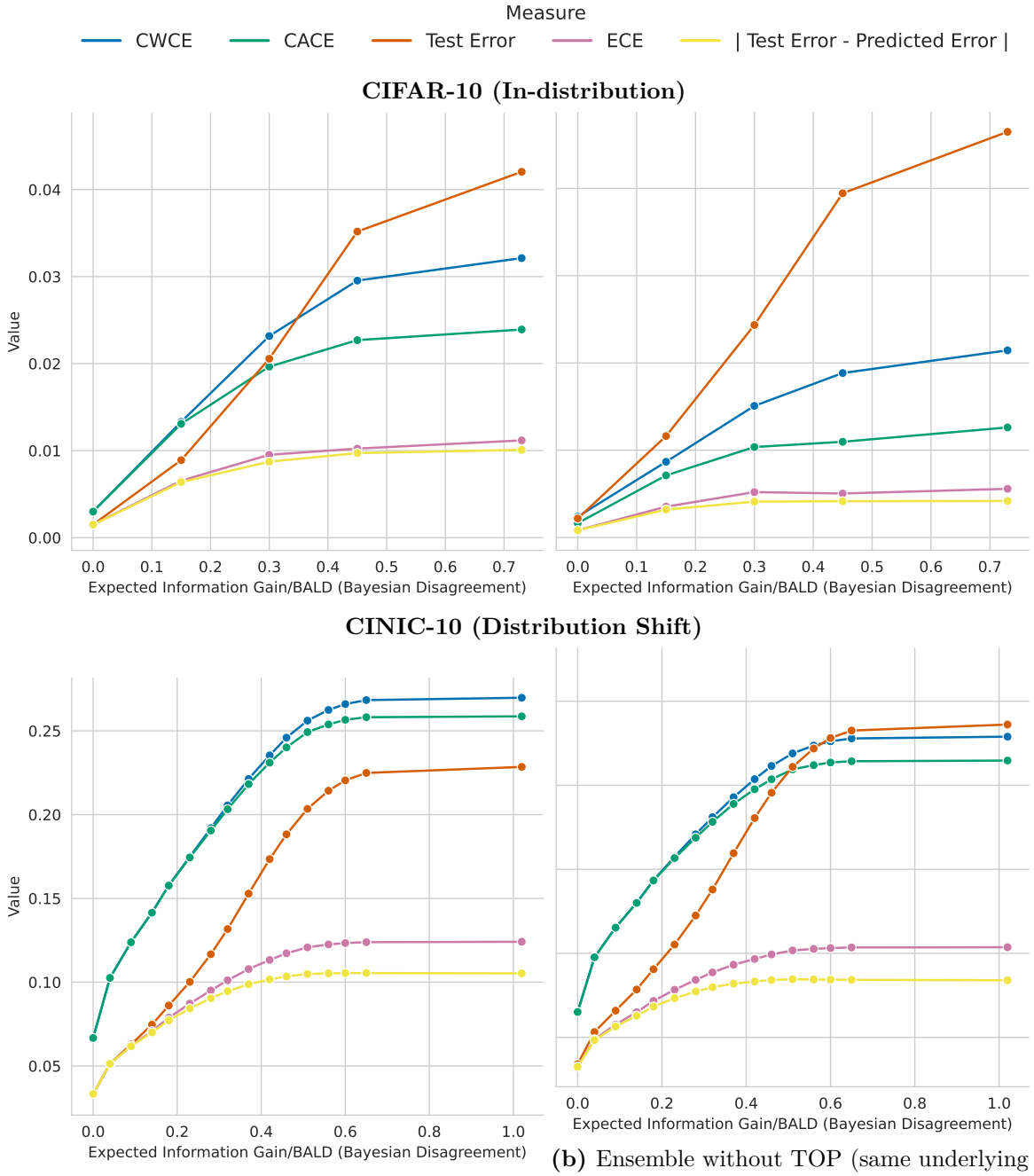
Additionally, in Figure B.7 and Figure B.8, we also show rejection plots using the Expected Information Gain/BALD for thresholding. We observe similar trajectories. Comparing these results with Figure B.5 and Figure B.6, we see that both the predicted error and the Bayesian metric behave similarly. We hypothesize that this could be because the datasets only contain few samples with high aleatoric uncertainty (e.g. noise), which would otherwise act as confounder [Mukhoti et al., 2023]. See also the discussion in §B.2.5.



(a) Ensemble with TOP

(b) Ensemble without TOP (same underlying models)

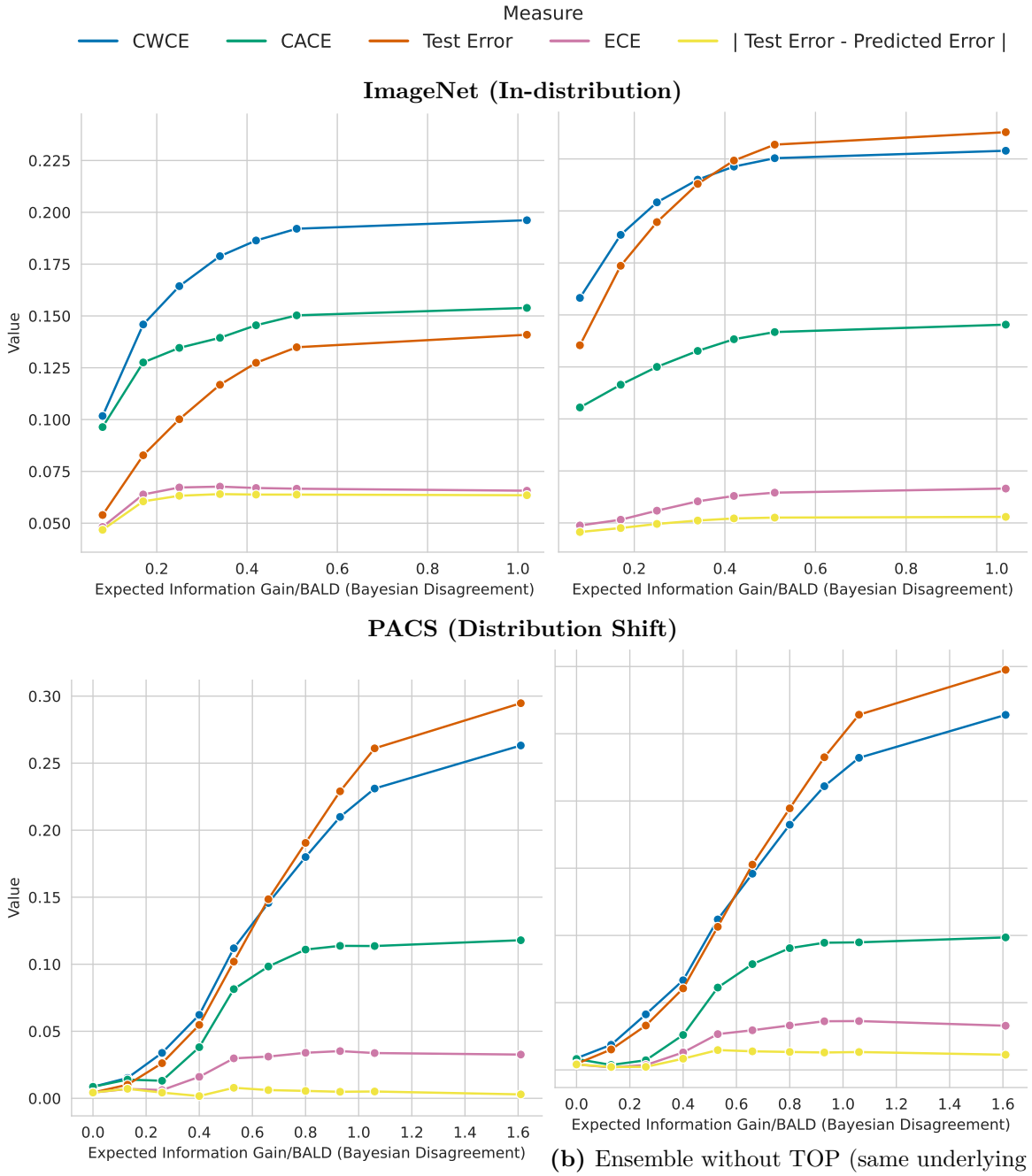
Figure B.6: *Rejection Plot of Calibration Metrics for Increasing Disagreement In-Distribution (ImageNet) and Under Distribution Shift (PACS 'photo' domain → other domains).* Different calibration metrics (*ECE*, CWCE, CACE) vary across ImageNet and PACS' 'art painting', 'cartoon', and 'sketch' domains across an ensemble of 5 models trained on ImageNet and 25 models fine-tuned on PACS' 'photo' domain, depending on the rejection threshold of the predicted error (disagreement rate). Again, calibration cannot be assumed constant for in-distribution data or under distribution shift. The mean predicted error (disagreement rate) is shown on the x-axis. **(a)** shows results for an ensemble using TOP (following Jiang et al. [2022]), and **(b)** for a regular deep ensemble without TOP. Details in §B.2.7.5.



(a) Ensemble with TOP

(b) Ensemble without TOP (same underlying models)

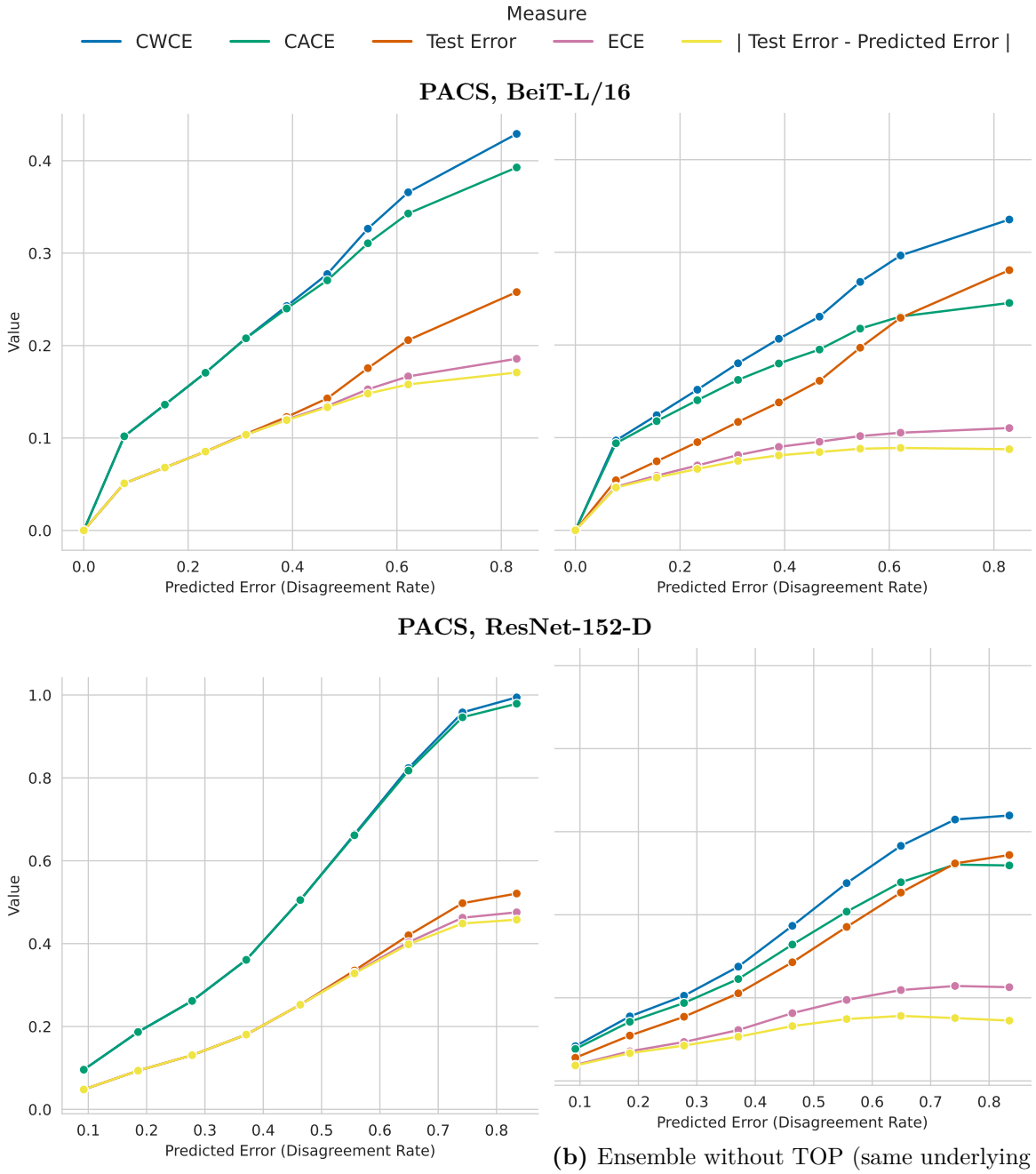
Figure B.7: *Rejection Plot of Calibration Metrics for Increasing Bayesian Disagreement In-Distribution (CIFAR-10) and Under Distribution Shift (CINIC-10).* Different calibration metrics (*ECE*, CWCE, CACE) vary across CIFAR-10 and CINIC-10, depending on the rejection threshold of Bayesian disagreement (Expected Information Gain/BALD). The trajectory matches the one for prediction disagreement. We hypothesize this is because there are few noisy samples in the dataset which would act as a confounder for prediction disagreement otherwise. Details in §B.2.7.5.



(a) Ensemble with TOP

(b) Ensemble without TOP (same underlying models)

Figure B.8: *Rejection Plot of Calibration Metrics for Increasing Bayesian Disagreement In-Distribution (CIFAR-10) and Under Distribution Shift (CINIC-10).* Different calibration metrics (*ECE*, CWCE, CACE) vary across CIFAR-10 and CINIC-10, depending on the rejection threshold of Bayesian disagreement (Expected Information Gain/BALD). The trajectory matches the one for prediction disagreement. We hypothesize this is because there are few noisy samples in the dataset which would act as a confounder for prediction disagreement otherwise. Details in §B.2.7.5.



(a) Ensemble with TOP

(b) Ensemble without TOP (same underlying models)

Figure B.9: *Rejection Plot of Calibration Metrics for Increasing Disagreement Under Distribution Shift (PACS 'photo' domain → other domains) for Specific Model Architectures.* We use the same encoder weights and evaluate on an ensemble of 5 models, which were last-layer fine-tuned on PACS. We show ResNet-152-D and BeiT-L/16. Details in §B.2.7.5.

B.3 Dirichlet Model of a Deep Ensemble's Softmax Predictions

How well can these distributions capture the outputs of an ensemble? Several works have examined distilling the uncertainty of deep ensembles into a single model [Malinin et al., 2019; Fathullah et al., 2021; Ryabinin et al., 2021] or using Dirichlet distributions to model epistemic uncertainty [Malinin and Gales, 2018, 2019; Hammam et al., 2022]. But In this chapter, we qualitatively examine how well Dirichlet distributions can capture predictions of deep ensembles in computer vision.

Concretely, we examine how well they can approximate the variance of softmax entropies when matching the predictions and epistemic uncertainty of input samples on OoD data (which is arguably more difficult than when trying to capture uncertainty of iD which will have low epistemic uncertainty). This is easier to visualize for datasets than individual predictions.

We qualitatively evaluate deep ensembles across different model architectures and qualitatively find that the modelled Dirichlet distributions provide more concentrated predictions. Hence, samples from them are also unlikely to model the actual individual predictions of a deep ensemble well. This does not invalidate any of the results in the previous chapters, but it does suggest that the Dirichlet distribution is not a good approximation of the posterior predictive distribution of the ensemble members.

B.3.1 Methodology

There are two interpretations of the ensemble parameter distribution $\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$:

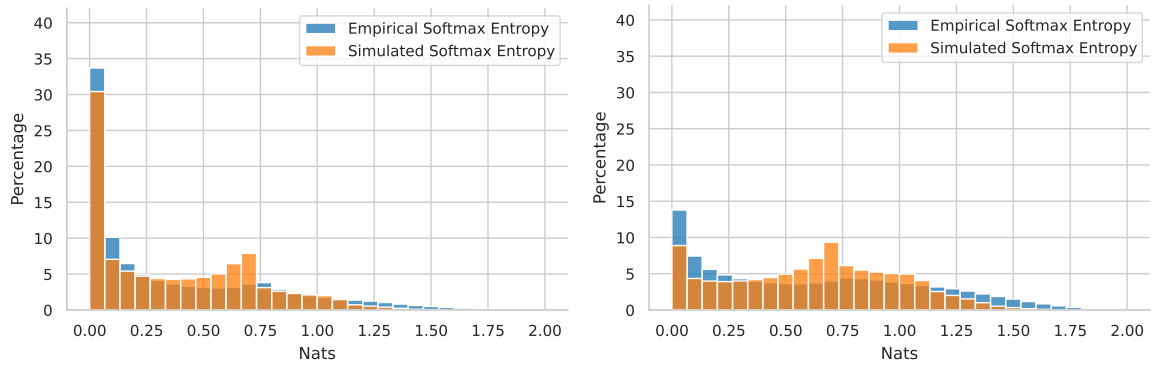
1. we can view it as an empirical distribution given a specific ensemble with members $\omega_{i\in\{1,\dots,K\}}$, or
2. we can view it as a distribution over all possible trained models which depends on random weight initializations, the dataset, stochasticity in the minibatches, and the optimization process.

In the latter case, any deep ensemble with K members can be seen as finite Monte-Carlo sample of this posterior distribution. The predictions of an ensemble then are an unbiased estimate of the predictive distribution $\mathbb{E}_{\mathrm{p}(\omega\mid\mathcal{D}^{\text{train}})}[\mathrm{p}(y \mid \boldsymbol{x}, \omega)]$, and similarly the expected information gain computed using the members of the deep ensemble is just a (biased) estimator of $\mathrm{I}[Y; \Omega \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$.

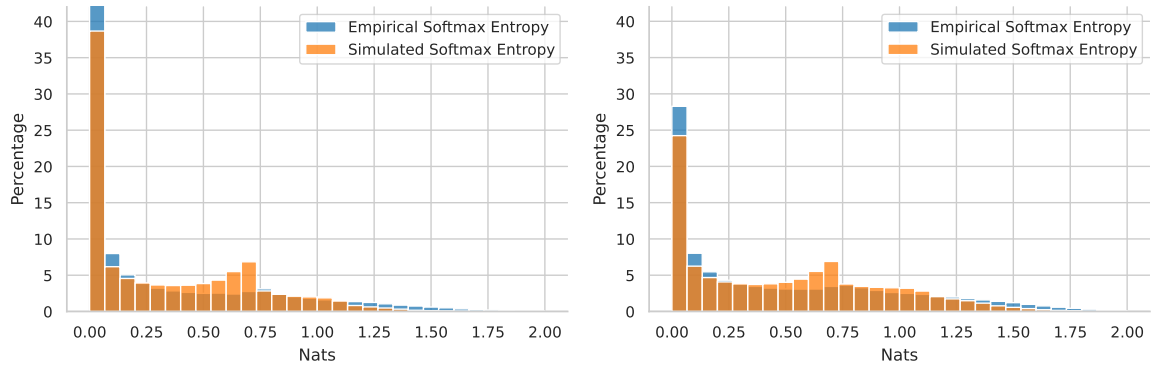
Inverse Problem. Based on this interpretation of deep ensembles as a distribution over model parameters, we can look at the following inverse problem: given *some value* for the predictive distribution and epistemic uncertainty of a deep ensemble, estimate what the softmax entropies from each ensemble component must have been. That is if we observe deep ensembles to have high epistemic uncertainty on (near) OoD data, we can deduce from that what the distribution of softmax entropy of deterministic neural nets (the ensemble members) ought to look like.

That is, given a predictive distribution $\mathrm{p}(y \mid x, \mathcal{D}^{\text{train}})$ and epistemic uncertainty $\mathrm{I}[Y; \Omega \mid x, \mathcal{D}^{\text{train}}]$ (expected information gain) of the deep ensemble, we can observe the softmax entropy $\mathrm{H}[Y \mid x, \omega]$ as a random variable of a single deterministic model, $\omega \sim \mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$, and estimate its variance $\mathrm{Var}_{\omega\mid\mathcal{D}^{\text{train}}}[\mathrm{H}[Y \mid x, \omega]]$.

Empirically, we find the real variance to be higher by a large amount for OoD samples, showing that softmax entropies do not capture epistemic uncertainty well for samples with high epistemic uncertainty.



(a) SVHN (OoD): VGG16 WideResNet-28-10+SN



(b) CIFAR-100 (OoD): VGG16 WideResNet-28-10+SN

Figure B.10: *Simulated vs Empirical Softmax Entropy and Predictive Entropy.* WideResNet-28-10+SN and VGG16 trained on CIFAR-10 (25 models). Although we use a simple Dirichlet model, sampling from the fitted Dirichlet distributions does approximate the empirical entropy distribution of softmax entropies well.

We will need to make several strong assumptions that limit the generality of our estimation, but we can show that our analysis models the resulting softmax entropy distributions appropriately. This will show that deterministic softmax models can have widely different entropies and confidence values.

Approximate Model. For a *fixed* $\boldsymbol{x}$, we approximate the distribution over softmax probability vectors $\mathrm{p}(y \mid \boldsymbol{x}, \omega)$ for different ω using a Dirichlet distribution $\mathbf{p} \sim \mathrm{Dir}(\alpha)$ with non-negative concentration parameters $\alpha = (\alpha_1, \ldots, \alpha_K)$ and $\alpha_0 := \sum \alpha_i$. Note that we only use the Dirichlet distribution *as an analysis tool*.

Concretely, for a distribution over models $\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$, and a sample $\boldsymbol{x}$, we obtain $\mathrm{p}(y \mid x, \mathcal{D}^{\text{train}})$, and $\mathrm{I}[Y; \Omega \mid x, \mathcal{D}^{\text{train}}]$. We use moment matching with these two quantities to fit a Dirichlet distribution $\mathbf{p} \sim \mathrm{Dir}(\alpha)$ on $\mathrm{p}(y \mid x, \omega)$ over Ω, which satisfies:

$$\mathrm{p}(y \mid x, \mathcal{D}^{\text{train}}) = \frac{\alpha_i}{\alpha_0} \tag{B.131}$$

$$\mathrm{H}[Y \mid x, \mathcal{D}^{\text{train}}] - \mathrm{I}[Y; \Omega \mid x, \mathcal{D}^{\text{train}}] = \psi(\alpha_0 + 1) - \sum_{y=1}^{K} \mathrm{p}(y \mid x) \psi(\alpha_0 \, \mathrm{p}(y \mid x) + 1). \tag{B.132}$$

Then, we can model the softmax distribution as given in eq. (B.20). The details and proofs can be found below in §B.3.4.

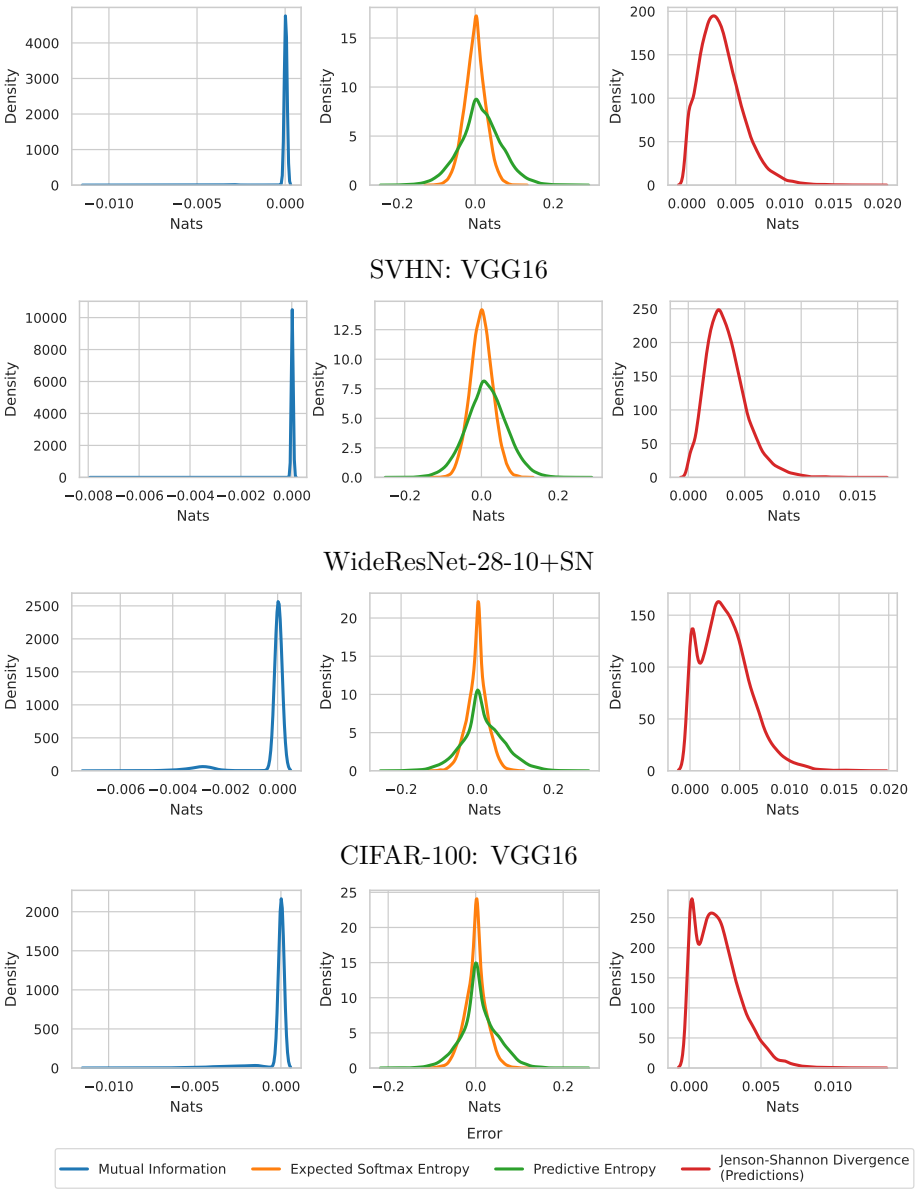
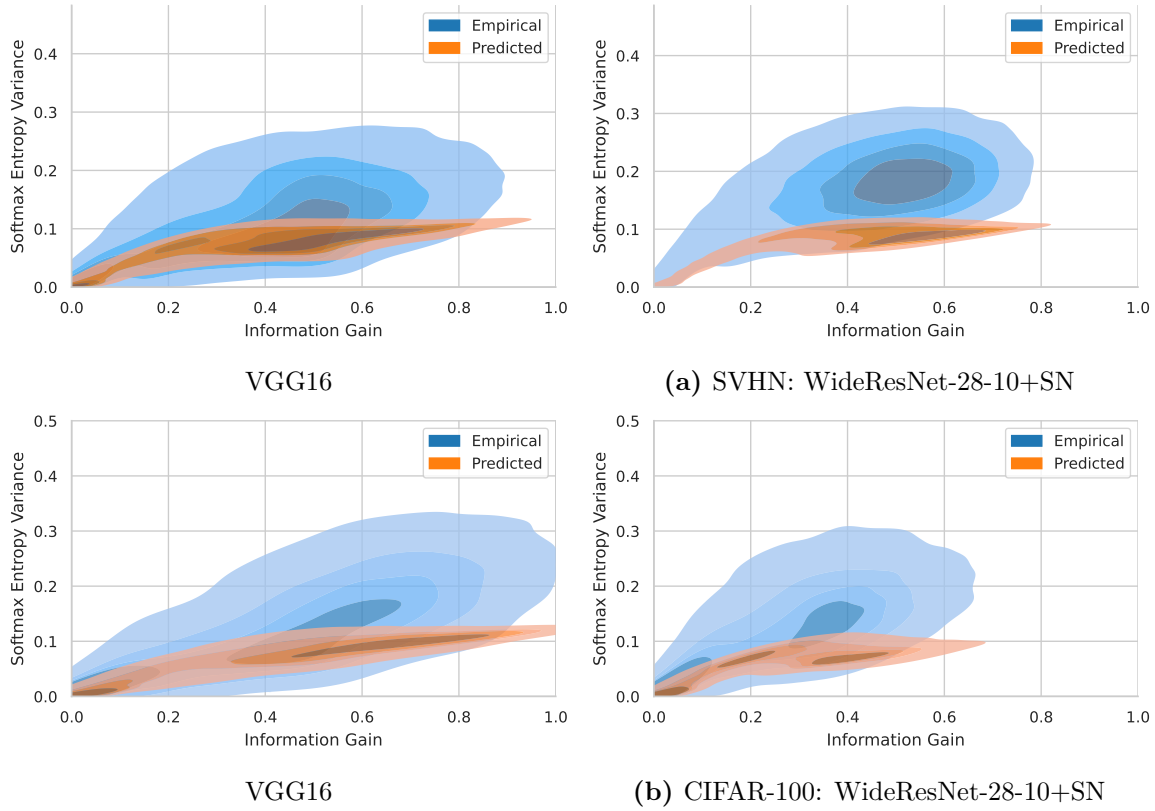


Figure B.11: *Simulated Quantities (via Dirichlet Distributions) vs Empirical Quantities.* WideResNet-28-10+SN and VGG16 trained on CIFAR-10 (25 models). Although we use a simple Dirichlet model, sampling from the fitted Dirichlet distributions does approximate the empirical entropy distribution of softmax entropies well.

B.3.2 Qualitative Empirical Validation

We train two deep ensembles of VGG and WideResNet-28-10+SN models (25 members each) on CIFAR-10 and compute the predictive entropy, mutual information, and softmax entropies for each sample in SVHN and CIFAR-100, which we use as OoD distribution. Then we fit a Dirichlet distribution on the softmax entropies of the ensemble members and use the fitted distribution to simulate the softmax entropies for the OoD samples. We fit the Dirichlet distribution on the BMA prediction of the ensemble and the corresponding mutual information (epistemic uncertainty).

We have already empirically verified that softmax entropies vary considerably in



VGG16 **(a)** SVHN: WideResNet-28-10+SN

VGG16 **(b)** CIFAR-100: WideResNet-28-10+SN

Figure B.12: *Simulated & empirical softmax entropy vs mutual information (EIG) on WideResNet-28-10+SN and VGG16.* Although we use a simple Dirichlet model, samples from the fitted Dirichlet distributions approximate the three major information quantities and the BMA of the ensemble predictions very well.

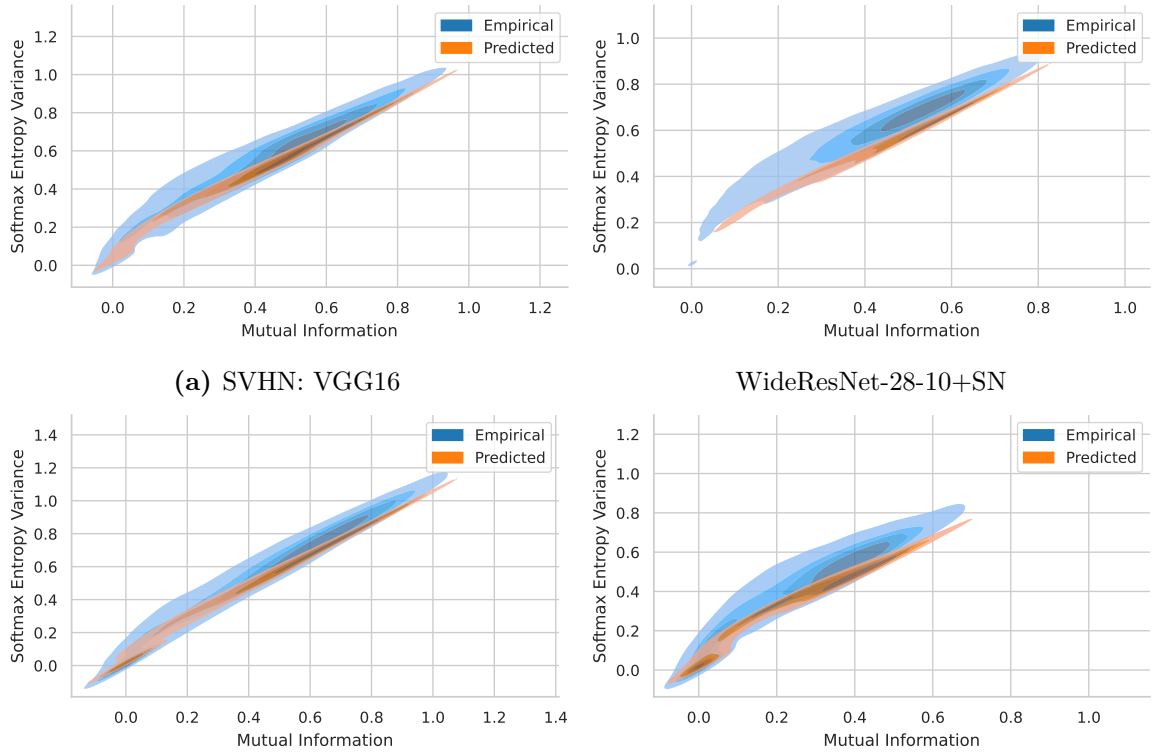
§3.1.3. The distribution of actual softmax entropies and predicted softmax entropies seems to be close in the histograms of Figure B.10, and the overall error terms for the three information quantities as well as the Jensen-Shannon Divergence between the simulated BMA predictions and real ones are small Figure B.11. Yet, Figures B.12 and B.13 show that the distributions are quite different in terms of softmax entropy variance and RMSE (for the softmax entropies as estimates of the respective predictive entropy).

B.3.3 Discussion

Hence, we can conclude without evaluating the simulated predictions themselves in more detail that the Dirichlet model is not a very good fit for the empirical distribution of softmax entropies from individual members of a deep ensemble. This does not invalidate that these approximations have great and useful value on downstream tasks, but it raises the question for future work to find distributions that can better capture the empirical distribution of softmax entropies from individual members of a deep ensemble.

B.3.4 Details

Based on the interpretation of deep ensembles as a distribution over model parameters, we can walk backwards and, given *some value* for the predictive distribution and epistemic uncertainty of a deep ensemble, estimate what the softmax entropies from each ensemble component must have been. I.e. if we observe deep ensembles to have



(a) SVHN: VGG16 WideResNet-28-10+SN

Figure B.13: *RMSE for simulated & empirical softmax entropy vs mutual information (EIG) on WideResNet-28-10+SN and VGG16.*

high epistemic uncertainty on OoD data, we can deduce from that what the softmax entropy of deterministic neural nets (the ensemble components) must look like. More specifically, given a predictive distribution $\mathrm{p}(y \mid x, \mathcal{D}^{\text{train}})$ and epistemic uncertainty, that is expected information gain $\mathrm{I}[Y; \Omega \mid x, \mathcal{D}^{\text{train}}]$, of the infinite deep ensemble, we estimate the expected softmax entropy from a single deterministic model, considered as a sample $\omega \sim \mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})$ and model the variance. Empirically, we find the real variance to be higher by a large amount for OoD samples, showing that softmax entropies do not capture epistemic uncertainty well for samples with high epistemic uncertainty. We will need to make several strong assumptions that limit the generality of our estimation.

Given the predictive distribution $\mathrm{p}(y \mid x, \mathcal{D}^{\text{train}})$ and epistemic uncertainty $\mathrm{I}[Y; \Omega \mid x, \mathcal{D}^{\text{train}}]$, we can approximate the distribution over softmax probability vectors $\mathrm{p}(y \mid \boldsymbol{x}, \omega)$ for different ω using its maximum-entropy estimate: a Dirichlet distribution $(Y_1, \ldots, Y_K) \sim \mathrm{Dir}(\alpha)$ with non-negative concentration parameters $\alpha = (\alpha_1, \ldots, \alpha_K)$ and $\alpha_0 := \sum \alpha_i$. Note that the Dirichlet distribution is used *only as an analysis tool*, and at no point do we need to actually fit Dirichlet distributions to our data.

Preliminaries. Before we can establish our main result, we need to look more closely at Dirichlet-Multinomial distributions. Given a Dirichlet distribution $\mathrm{Dir}(\alpha)$ and a random variable $\mathbf{p} \sim \mathrm{Dir}(\alpha)$, we want to quantify the expected entropy $\mathbb{E}_{\mathbf{p} \sim \mathrm{Dir}(\alpha)} \mathrm{H}_{Y \sim \mathrm{Cat}(\mathbf{p})}[Y]$ and its variance $\mathrm{Var}_{\mathbf{p} \sim \mathrm{Dir}(\alpha)}[\mathrm{H}_{Y \sim \mathrm{Cat}(\mathbf{p})}[Y]]$. For this, we need to develop more theory. In the following, Γ denotes the Gamma function, ψ denotes the Digamma function, ψ' denotes the Trigamma function.

Lemma B.16. *Given a Dirichlet distribution and random variable* $\mathbf{p} \sim \mathrm{Dir}(\alpha)$*, the following hold:*

1. The expectation $\mathbb{E}[\log \mathbf{p}_i]$ *is given by:*

$$\mathbb{E}[\log \mathbf{p}_i] = \psi(\alpha_i) - \psi(\alpha_0). \tag{B.133}$$

2. The covariance $\mathrm{Cov}[\log \mathbf{p}_i, \log \mathbf{p}_j]$ *is given by*

$$\mathrm{Cov}[\log \mathbf{p}_i, \log \mathbf{p}_j] = \psi'(\alpha_i)\, \delta_{ij} - \psi'(\alpha_0). \tag{B.134}$$

3. The expectation $\mathbb{E}[\mathbf{p}_i^n \mathbf{p}_j^m \log \mathbf{p}_i]$ *is given by:*

$$\begin{aligned} &\mathbb{E}[\mathbf{p}_i^n \mathbf{p}_j^m \log \mathbf{p}_i] \\ &\quad = \frac{\alpha_i^{\overline{n}}\, \alpha_j^{\overline{m}}}{\alpha_0^{\overline{n+m}}} \left(\psi(\alpha_i + n) - \psi(\alpha_0 + n + m)\right), \end{aligned} \tag{B.135}$$

where $i \neq j$*, and* $n^{\overline{k}} = n\,(n+1)\,\ldots\,(n+k-1)$ *denotes the rising factorial.*

Proof. 1. The Dirichlet distribution is members of the exponential family. Therefore, the moments of the sufficient statistics are given by the derivatives of the partition function with respect to the natural parameters. The natural parameters of the Dirichlet distribution are just its concentration parameters α_i. The partition function is

$$A(\alpha) = \sum_{i=1}^{k} \log \Gamma\left(\alpha_i\right) - \log \Gamma\left(\alpha_0\right), \tag{B.136}$$

the sufficient statistics is $T(\boldsymbol{x}) = \log \boldsymbol{x}$, and the expectation $\mathbb{E}[T]$ is given by

$$\mathbb{E}[T_i] = \frac{\partial A(\alpha)}{\partial \alpha_i} \tag{B.137}$$

as the Dirichlet distribution is a member of the exponential family. Substituting the definitions and evaluating the partial derivative yields

$$\mathbb{E}[\log \mathbf{p}_i] = \frac{\partial}{\partial \alpha_i} \left[\sum_{i=1}^{k} \log \Gamma\left(\alpha_i\right) - \log \Gamma\left(\sum_{i=1}^{k} \alpha_i\right)\right] \tag{B.138}$$

$$= \psi\left(\alpha_i\right) - \psi\left(\alpha_0\right) \frac{\partial}{\partial \alpha_i} \alpha_0, \tag{B.139}$$

where we have used that the Digamma function ψ is the log derivative of the Gamma function $\psi(\boldsymbol{x}) = \frac{d}{dx} \ln \Gamma(\boldsymbol{x})$. This proves (B.133) as $\frac{\partial}{\partial \alpha_i} \alpha_0 = 1$.
2. Similarly, the covariance is obtained using a second-order partial derivative:

$$\mathrm{Cov}[T_i, T_j] = \frac{\partial^2 A(\alpha)}{\partial \alpha_i\, \partial \alpha_i}. \tag{B.140}$$

Again, substituting yields

$$\mathrm{Cov}[\log \mathbf{p}_i, \log \mathbf{p}_j] = \frac{\partial}{\partial \alpha_j} \left[\psi\left(\alpha_i\right) - \psi\left(\alpha_0\right)\right] \tag{B.141}$$

$$= \psi'\left(\alpha_i\right) \delta_{ij} - \psi'\left(\alpha_0\right). \tag{B.142}$$

3. We will make use of a simple reparameterization to prove the statement using Equation B.133. Expanding the expectation and substituting the density $\mathrm{Dir}(\mathbf{p};\alpha)$, we obtain

$$\mathbb{E}[\mathbf{p}_i^n \mathbf{p}_j^m \log \mathbf{p}_i] = \int \mathrm{Dir}(\mathbf{p};\alpha)\, \mathbf{p}_i^n\, \mathbf{p}_j^m\, \log \mathbf{p}_i \, d\mathbf{p} \tag{B.143}$$

$$= \int \frac{\Gamma(\alpha_0)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{k=1}^K \mathbf{p}_k^{\alpha_k - 1}\, \mathbf{p}_i^n\, \mathbf{p}_j^m\, \log \mathbf{p}_i \, d\mathbf{p} \tag{B.144}$$

$$= \frac{\Gamma(\alpha_i + n)\Gamma(\alpha_j + m)\Gamma(\alpha_0 + n + m)}{\Gamma(\alpha_i)\Gamma(\alpha_j)\Gamma(\alpha_0)} \int \mathrm{Dir}(\hat{\mathbf{p}};\hat{\alpha})\, \hat{\mathbf{p}}_i^n\, \hat{\mathbf{p}}_j^m\, \log \hat{\mathbf{p}}_i \, d\hat{\mathbf{p}} \tag{B.145}$$

$$= \frac{\alpha_i^{\overline{n}}\, \alpha_j^{\overline{m}}}{\alpha_0^{\overline{n+m}}}\, \mathbb{E}[\log \hat{\mathbf{p}}_i], \tag{B.146}$$

where $\hat{\mathbf{p}} \sim \mathrm{Dir}(\hat{\alpha})$ with $\hat{\alpha} = (\alpha_0, \ldots, \alpha_i + n, \ldots, \alpha_j + m, \ldots, \alpha_K)$, and we made use of the fact that $\frac{\Gamma(z+n)}{\Gamma(z)} = z^{\overline{n}}$. Finally, we can apply Equation B.133 on $\hat{\mathbf{p}} \sim \mathrm{Dir}(\hat{\alpha})$ to show

$$= \frac{\alpha_i^{\overline{n}}\, \alpha_j^{\overline{m}}}{\alpha_0^{\overline{n+m}}} \left(\psi(\alpha_i + n) - \psi(\alpha_0 + n + m)\right). \tag{B.147}$$

□

With this, we can already quantify the expected entropy $\mathbb{E}_{\mathbf{p}\sim\mathrm{Dir}(\alpha)}\, \mathrm{H}_{Y\sim\mathrm{Cat}(\mathbf{p})}[Y]$:

Lemma B.17. *Given a Dirichlet distribution and a random variable* $\mathbf{p} \sim Dir(\alpha)$*, the expected entropy* $\mathbb{E}_{\mathbf{p}\sim\mathrm{Dir}(\alpha)}\, \mathrm{H}_{Y\sim\mathrm{Cat}(\mathbf{p})}[Y]$ *of the categorical distribution* $Y \sim \mathrm{Cat}(\mathbf{p})$ *is given by*

$$\mathbb{E}_{\mathrm{p}(\mathbf{p}|\alpha)}\, \mathrm{H}[Y \mid \mathbf{p}] = \psi(\alpha_0 + 1) - \sum_{y=1}^K \frac{\alpha_i}{\alpha_0} \psi(\alpha_i + 1). \tag{B.148}$$

Proof. Applying the sum rule of expectations and Equation B.135 from Lemma B.16, we can write

$$\mathbb{E}\, \mathrm{H}[Y \mid \mathbf{p}] = \mathbb{E}[-\sum_{i=1}^K \mathbf{p}_i \log \mathbf{p}_i] = -\sum_i \mathbb{E}[\mathbf{p}_i \log \mathbf{p}_i] \tag{B.149}$$

$$= -\sum_i \frac{\alpha_i}{\alpha_0} \left(\psi(\alpha_i + 1) - \psi(\alpha_0 + 1)\right). \tag{B.150}$$

The result follows after rearranging and making use of $\sum_i \frac{\alpha_i}{\alpha_0} = 1$. □

With these statements, we can answer a slightly more complex problem:

Lemma B.18. *Given a Dirichlet distribution and a random variable* $\mathbf{p} \sim \mathrm{Dir}(\alpha)$*, the covariance* $\mathrm{Cov}[\mathbf{p}_i^n \log \mathbf{p}_i, \mathbf{p}_j^m \log \mathbf{p}_j]$ *is given by*

$$\mathrm{Cov}[\mathbf{p}_i^n \log \mathbf{p}_i, \mathbf{p}_j^m \log \mathbf{p}_j] \tag{B.151}$$

$$
\begin{aligned}
&= \frac{\alpha_i^{\overline{n}}\,\alpha_j^{\overline{m}}}{\alpha_0^{\overline{n+m}}}\,((\psi(\alpha_i+n)-\psi(\alpha_0+n+m))\\
&\quad (\psi(\alpha_j+m)-\psi(\alpha_0+n+m))\\
&\quad -\psi'(\alpha_0+n+m))\\
&\quad + \frac{\alpha_i^{\overline{n}}\,\alpha_j^{\overline{m}}}{\alpha_0^{\overline{n}}\,\alpha_0^{\overline{m}}}(\psi(\alpha_i+n)-\psi(\alpha_0+n))\\
&\quad (\psi(\alpha_j+m)-\psi(\alpha_0+n)),
\end{aligned}
\tag{B.152}
$$

for $i \neq j$, where ψ is the Digamma function and ψ' is the Trigamma function. Similarly, the covariance $\mathrm{Cov}[\mathbf{p}_i^n \log \mathbf{p}_i, \mathbf{p}_i^m \log \mathbf{p}_i]$ *is given by*

$$
\mathrm{Cov}[\mathbf{p}_i^n \log \mathbf{p}_i, \mathbf{p}_i^m \log \mathbf{p}_i] \tag{B.153}
$$

$$
\begin{aligned}
&= \frac{\alpha_i^{\overline{n+m}}}{\alpha_0^{\overline{n+m}}}\Big((\psi(\alpha_i+n+m)-\psi(\alpha_0+n+m))^2\\
&\quad + \psi'(\alpha_i+n+m)-\psi'(\alpha_0+n+m))\\
&\quad + \frac{\alpha_i^{\overline{n}}\,\alpha_i^{\overline{m}}}{\alpha_0^{\overline{n}}\,\alpha_0^{\overline{m}}}(\psi(\alpha_i+n)-\psi(\alpha_0+n))\\
&\qquad (\psi(\alpha_i+m)-\psi(\alpha_0+n)).
\end{aligned}
\tag{B.154}
$$

Regrettably, the equations are getting large. By abuse of notation, we introduce a convenient shorthand before proving the lemma.

Definition B.8. We will denote by

$$
\overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{n,m}]} = \psi(\alpha_i+n)-\psi(\alpha_0+n+m), \tag{B.155}
$$

and use $\overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{n}]}$ for $\overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{n,0}]}$. Likewise,

$$
\overline{\mathrm{Cov}[\log \hat{\mathbf{p}}_i^{n,m}, \log \hat{\mathbf{p}}_j^{n,m}]} = \psi'(\alpha_i+n)\delta_{ij} - \psi'(\alpha_0+n+m). \tag{B.156}
$$

This notation agrees with the proof of Equation B.133 and (B.134) in Lemma B.16. With this, we can significantly simplify the previous statements:

Corollary B.19. *Given a Dirichlet distribution and random variable* $\mathbf{p} \sim \mathrm{Dir}(\alpha)$*,*

$$
\mathbb{E}[\mathbf{p}_i^n \mathbf{p}_j^m \log \mathbf{p}_i] = \frac{\alpha_i^{\overline{n}}\,\alpha_j^{\overline{m}}}{\alpha_0^{\overline{n+m}}}\overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{n,m}]}, \tag{B.157}
$$

$$
\mathrm{Cov}[\mathbf{p}_i^n \log \mathbf{p}_i, \mathbf{p}_j^m \log \mathbf{p}_j] \tag{B.158}
$$

$$
\begin{aligned}
&= \frac{\alpha_i^{\overline{n}}\,\alpha_j^{\overline{m}}}{\alpha_0^{\overline{n+m}}}\Big(\overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{n,m}]\mathbb{E}[\log \hat{\mathbf{p}}_j^{m,n}]}\\
&\qquad \overline{\mathrm{Cov}[\log \hat{\mathbf{p}}_i^{n,m}, \log \hat{\mathbf{p}}_j^{n,m}]}\Big)
\end{aligned}
\tag{B.159}
$$

$$
+ \frac{\alpha_i^{\overline{n}}\,\alpha_j^{\overline{m}}}{\alpha_0^{\overline{n}}\,\alpha_0^{\overline{m}}}\overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{n}]\mathbb{E}[\log \hat{\mathbf{p}}_j^{m}]} \quad \textit{for } i \neq j, \textit{ and}
$$

$$
\mathrm{Cov}[\mathbf{p}_i^n \log \mathbf{p}_i, \mathbf{p}_i^m \log \mathbf{p}_i] \tag{B.160}
$$

$$= \frac{\alpha_i^{\overline{n+m}}}{\alpha_0^{\overline{n+m}}} \Big(\overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{n+m}]}^2 + \overline{\mathrm{Cov}[\log \hat{\mathbf{p}}_i^{n+m}, \log \hat{\mathbf{p}}_i^{n+m}]} \Big) + \frac{\alpha_i^{\overline{n}}\, \alpha_i^{\overline{m}}}{\alpha_0^{\overline{n}}\, \alpha_0^{\overline{m}}} \overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{n}] \mathbb{E}[\log \hat{\mathbf{p}}_j^{m}]}. \tag{B.161}$$

Proof of Lemma B.18. This proof applies the well-know formula **(cov)** $\mathrm{Cov}[X, Y] = \mathbb{E}[X\,Y] - \mathbb{E}[X]\,\mathbb{E}[Y]$ once forward and once backward **(rcov)** $\mathbb{E}[X\,Y] = \mathrm{Cov}[X, Y] + \mathbb{E}[X]\,\mathbb{E}[Y]$ while applying Equation B.135 several times:

$$\mathrm{Cov}[\mathbf{p}_i^n \log \mathbf{p}_i, \mathbf{p}_j^m \log \mathbf{p}_j] \tag{B.162}$$

$$\overset{\textbf{cov}}{=} \mathbb{E}[\mathbf{p}_i^n \log(\mathbf{p}_i)\, \mathbf{p}_j^m \log(\mathbf{p}_j)] - \mathbb{E}[\mathbf{p}_i^n \log \mathbf{p}_i]\, \mathbb{E}[\mathbf{p}_j^m \log \mathbf{p}_j] \tag{B.163}$$

$$= \frac{\alpha_i^{\overline{n}}\, \alpha_j^{\overline{m}}}{\alpha_0^{\overline{n+m}}} \mathbb{E}[\log(\hat{\mathbf{p}}_i^{i,j}) \log(\hat{\mathbf{p}}_j^{i,j})] - \mathbb{E}[\log \hat{\mathbf{p}}_i^{i}]\, \mathbb{E}[\log \mathbf{p}_j^{j}] \tag{B.164}$$

$$\overset{\textbf{(rcov)}}{=} \frac{\alpha_i^{\overline{n}}\, \alpha_j^{\overline{m}}}{\alpha_0^{\overline{n+m}}} \Big(\mathrm{Cov}[\log \hat{\mathbf{p}}_i^{i,j}, \log \hat{\mathbf{p}}_j^{i,j}] + \mathbb{E}[\log \hat{\mathbf{p}}_i^{i,j}]\, \mathbb{E}[\log \hat{\mathbf{p}}_j^{i,j}] \Big) - \frac{\alpha_i^{\overline{n}}\, \alpha_j^{\overline{m}}}{\alpha_0^{\overline{n}}\, \alpha_0^{\overline{m}}} \mathbb{E}[\log \hat{\mathbf{p}}_i^{i}]\, \mathbb{E}[\log \mathbf{p}_j^{j}], \tag{B.165}$$

where $\mathbf{p}^{i,j} \sim \mathrm{Dir}(\alpha^{i,j})$ with $\alpha^{i,j} = (\ldots, \alpha_i + n, \ldots, \alpha_j + m, \ldots)$. $\mathbf{p}^{i/j}$ and $\alpha^{i/j}$ are defined analogously. Applying Equation B.134 and Equation B.133 from Lemma B.16 yields the statement. For $i = j$, the proof follows the same pattern. □

Variance of Softmax Entropy Now, we can prove the theorem that quantifies the variance of the entropy of Y:

Theorem B.20. *Given a Dirichlet distribution and a random variable* $\mathbf{p} \sim Dir(\alpha)$*, the variance of the entropy* $\mathrm{Var}_{\mathbf{p}\sim\mathrm{Dir}(\alpha)}[\mathrm{H}_{Y\sim\mathrm{Cat}(\mathbf{p})}[Y]]$ *of the categorical distribution* $Y \sim \mathrm{Cat}(\mathbf{p})$ *is given by*

$$\mathrm{Var}[\mathrm{H}[Y \mid \mathbf{p}]] \tag{B.166}$$

$$= \sum_i \frac{\alpha_i^{\overline{2}}}{\alpha_0^{\overline{2}}} \Big(\overline{\mathrm{Cov}[\log \hat{\mathbf{p}}_i^2, \log \hat{\mathbf{p}}_i^2]} + \overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^2]}^2 \Big) + \sum_{i \neq j} \frac{\alpha_i\, \alpha_j}{\alpha_0^{\overline{2}}} \Big(\overline{\mathrm{Cov}[\log \hat{\mathbf{p}}_i^1, \log \hat{\mathbf{p}}_j^1]} + \overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^{1,1}]}\, \overline{\mathbb{E}[\log \hat{\mathbf{p}}_j^{1,1}]} \Big) - \sum_{i,j} \frac{\alpha_i\, \alpha_j}{\alpha_0^2} \overline{\mathbb{E}[\log \hat{\mathbf{p}}_i^1] \mathbb{E}[\log \hat{\mathbf{p}}_j^1]}. \tag{B.167}$$

Proof. We start by applying the well-known formula $\mathrm{Var}[\sum_i X_i] = \sum_{i,j} \mathrm{Cov}[X_i, X_j]$ and then apply Lemma B.18 repeatedly. □

C

Single Forward-Pass Aleatoric and Epistemic Uncertainty

C.1 Experimental Details

C.1.1 Dirty-MNIST

We train for 50 epochs using SGD with a momentum of 0.9 and an initial learning rate of 0.1. The learning rate drops by a factor of 10 at training epochs 25 and 40. Following SNGP [Liu et al., 2020a], we apply online spectral normalization with one step of a power iteration on the convolutional weights. For 1x1 convolutions, we use the exact algorithm, and for 3x3 convolutions, the approximate algorithm from Gouk et al. [2021]. The coefficient for SN is a hyperparameter which we set to 3 using cross-validation.

C.1.2 OoD Detection Training Setup

We train the softmax baselines on CIFAR-10/100 for 350 epochs using SGD as the optimizer with a momentum of 0.9, and an initial learning rate of 0.1. The learning rate drops by a factor of 10 at epochs 150 and 250. We train the 5-Ensemble baseline using this same training setup. The SNGP and DUQ models were trained using the setup of SNGP and hyperparameters mentioned in their respective papers [Liu et al., 2020a; van Amersfoort et al., 2020]. For models trained on ImageNet, we train for 90 epochs with SGD optimizer, an initial learning rate of 0.1 and a weight decay of 1e-4. We use a learning rate warm-up decay of 0.01 along with a step scheduler with step size of 30 and a step factor of 0.1.

C.1.3 Semantic Segmentation Training Setup

In Figure C.1, we plot the L2 distance between feature space means of different classes for a pair of randomly chosen distant pixels on the Pascal VOC 2012 val set. We observe that feature space means between pairs of different classes are more distant compared to the same class irrespective of the location of the pixel for the class. This leads us to construct a Gaussian mean and covariance matrix per class as opposed to one mean and one covariance matrix per class per pixel, thereby greatly reducing the computational load of fitting a GMM in semantic segmentation. Similar to classification, we treat each pixel in the training set as a separate sample and fit a single Gaussian mean and covariance matrix per class.

For the semantic segmentation experiment, we use a DeepLab-v3+ [Chen et al., 2017] model with a ResNet-101 backbone as the architecture of choice. We train each of the models on Pascal VOC for 50 epochs using SGD as the optimizer, with

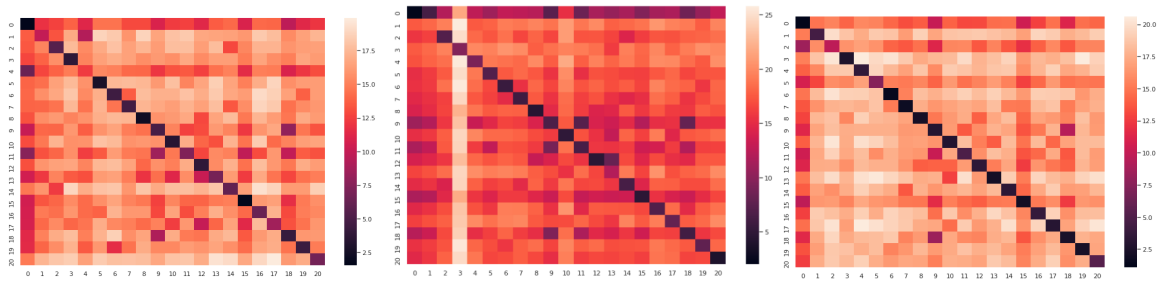


Figure C.1: L2 distances between the feature space means of different classes for a pair of distant pixels on the Pascal VOC 2012 val set: (left) Pixels $(10, 255)$ and $(500, 255)$, (middle) Pixels $(234, 349)$ and $(36, 22)$ and (right) Pixels $(300, 500)$ and $(400, 255)$.

a momentum of 0.9 and a weight decay of $5e-4$. We set the initial learning rate to 0.007 with a polynomial decay during the course of training. Finally, we trained with a batch size of 32 parallelized over 4 GPUs.

C.1.4 Compute Resources

Each model (ResNet-18, Wide-ResNet-28-10, ResNet-50, ResNet-110, DenseNet-121 or VGG-16) used for the large scale active learning, CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C and CIFAR-100 vs SVHN/Tiny-ImageNet tasks was trained on a single Nvidia Quadro RTX 6000 GPU. Each model (LeNet, VGG-16 and ResNet-18) used to get the results in Figure 3.2 and Table 3.4 was trained on a single Nvidia GeForce RTX 2060 GPU. Each model (ResNet-50, Wide-ResNet-50-2, VGG-16) trained on ImageNet was trained using 8 Nvidia Quadro RTX 6000 GPUs.

C.2 Additional Results

In this section, we provide details of additional results on the OoD detection task using CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C and CIFAR-100 vs SVHN/Tiny-ImageNet for ResNet-50, ResNet-110 and DenseNet-121 architectures. We present results on ResNet-50, ResNet-110 and DenseNet-121 for CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet in Table C.1, Table C.2 and Table C.3 respectively. We also present results on individual corruption types for CIFAR-10-C for Wide-ResNet-28-10, ResNet-50, ResNet-110 and DenseNet-121 in Figure C.2, Figure C.3, Figure C.4 and Figure C.5 respectively.

Finally, we provide results for various ablations on DDU. As mentioned in §3.4, DDU consists of a deterministic softmax model trained with appropriate inductive biases. It uses softmax entropy to quantify aleatoric uncertainty and feature-space density to quantify epistemic uncertainty. In the ablation, we try to experimentally evaluate the following scenarios:

1. **Effect of inductive biases (sensitivity + smoothness):** We want to see the effect of removing the proposed inductive biases (i.e. no sensitivity and smoothness constraints) on the OoD detection performance of a model. To do this, we train a VGG-16 with and without spectral normalization. Note that VGG-16 does not have residual connections and hence, a VGG-16 does not follow the sensitivity and smoothness (bi-Lipschitz) constraints.

Table C.1: *OoD detection performance of different baselines using a ResNet-50 architecture with the CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet dataset pairs averaged over 25 runs.* Note: SN stands for Spectral Normalization, JP stands for Jacobian Penalty. We highlight the best deterministic and best method overall in bold for each metric.

Train Dataset	**Method**	**Penalty**	Aleatoric Uncertainty	**Epistemic Uncertainty**	Accuracy (↑)	ECE (↓)	**AUROC**		
							SVHN (↑)	**CIFAR-100 (↑)**	**Tiny-ImageNet (↑)**
CIFAR-10	Softmax	-	Softmax Entropy	Softmax Entropy	**95.04 ± 0.05**	**0.97 ± 0.04**	93.80 ± 0.41	88.91 ± 0.07	88.32 ± 0.07
	Energy-based [Liu et al., 2020b]	-		Softmax Density			94.48 ± 0.44	88.84 ± 0.08	88.45 ± 0.08
	DUQ [van Amersfoort et al., 2020]	JP	Kernel Distance	Kernel Distance	94.05 ± 0.11	1.71 ± 0.07	93.14 ± 0.43	83.87 ± 0.27	84.28 ± 0.26
	SNGP [Liu et al., 2020a]	SN	Predictive Entropy	Predictive Entropy	94.90 ± 0.11	1.01 ± 0.03	93.15 ± 0.85	89.32 ± 0.10	88.96 ± 0.13
	DDU (ours)	SN	Softmax Entropy	GMM Density	94.92 ± 0.06	1 ± 0.04	**94.77 ± 0.35**	**89.98 ± 0.17**	**89.12 ± 0.13**
	5-Ensemble [Lakshminarayanan et al., 2017]	-	Predictive Entropy	Predictive Entropy	**96.06 ± 0.04**	1.65 ± 0.07	94.75 ± 0.39	89.87 ± 0.06	88.69 ± 0.05
				Mutual Information			94.09 ± 0.20	89.76 ± 0.06	89.04 ± 0.03
					Test Accuracy (↑)	**Test ECE (↓)**	**SVHN (↑)**		**Tiny-ImageNet (↑)**
CIFAR-100	Softmax	-	Softmax Entropy	Softmax Entropy	77.91 ± 0.09	4.32 ± 0.10	81.32 ± 0.65		79.83 ± 0.07
	Energy-based [Liu et al., 2020b]	-		Softmax Density			82.05 ± 0.69		79.61 ± 0.08
	SNGP [Liu et al., 2020a]	SN	Predictive Entropy	Predictive Entropy	74.73 ± 0.22	7.68 ± 0.13	82.50 ± 2.09		77.05 ± 0.16
	DDU (ours)	SN	Softmax Entropy	GMM Density	**79.26 ± 0.16**	**4.07 ± 0.06**	**87.34 ± 0.64**		**82.11 ± 0.20**
	5-Ensemble [Lakshminarayanan et al., 2017]	-	Predictive Entropy	Predictive Entropy	**81.06 ± 0.07**	**3.54 ± 0.12**	83.42 ± 0.89		77.69 ± 0.12
				Mutual Information			84.24 ± 0.90		81.59 ± 0.05

Table C.2: *OoD detection performance of different baselines using a ResNet-110 architecture with the CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet dataset pairs averaged over 25 runs.* Note: SN stands for Spectral Normalization, JP stands for Jacobian Penalty. We highlight the best deterministic and best method overall in bold for each metric.

Train Dataset	**Method**	**Penalty**	Aleatoric Uncertainty	**Epistemic Uncertainty**	Accuracy (↑)	ECE (↓)	**AUROC**		
							SVHN (↑)	**CIFAR-100 (↑)**	**Tiny-ImageNet (↑)**
CIFAR-10	Softmax	-	Softmax Entropy	Softmax Entropy	**95.08 ± 0.04**	1.02 ± 0.04	93.12 ± 0.44	88.7 ± 0.1	88.07 ± 0.11
	Energy-based [Liu et al., 2020b]	-		Softmax Density			93.67 ± 0.47	88.60 ± 0.11	88.13 ± 0.11
	DUQ [van Amersfoort et al., 2020]	JP	Kernel Distance	Kernel Distance	94.32 ± 0.17	1.21 ± 0.07	94.02 ± 0.45	86.17 ± 0.35	85.24 ± 0.21
	SNGP [Liu et al., 2020a]	SN	Predictive Entropy	Predictive Entropy	94.85 ± 0.09	1.04 ± 0.02	93.17 ± 0.53	89.23 ± 0.10	88.80 ± 0.12
	DDU (ours)	SN	Softmax Entropy	GMM Density	94.82 ± 0.06	**1.01 ± 0.04**	**95.48 ± 0.30**	**90.08 ± 0.13**	**89.18 ± 0.15**
	5-Ensemble [Lakshminarayanan et al., 2017]	-	Predictive Entropy	Predictive Entropy	**96.18 ± 0.05**	1.57 ± 0.05	95.07 ± 0.45	**90.23 ± 0.04**	89 ± 0.03
				Mutual Information			94.72 ± 0.34	89.69 ± 0.05	88.35 ± 0.05
					Test Accuracy (↑)	**Test ECE (↓)**	**AUROC SVHN (↑)**		**AUROC Tiny-ImageNet (↑)**
CIFAR-100	Softmax	-	Softmax Entropy	Softmax Entropy	78.65 ± 0.10	3.93 ± 0.13	82.04 ± 0.57		80.13 ± 0.07
	Energy-based [Liu et al., 2020b]	-		Softmax Density			82.78 ± 0.60		80.01 ± 0.09
	SNGP [Liu et al., 2020a]	SN	Predictive Entropy	Predictive Entropy	76.16 ± 0.27	6.43 ± 0.75	83.94 ± 0.10		78.54 ± 0.28
	DDU (ours)	SN	Softmax Entropy	GMM Density	**78.89 ± 0.17**	**3.79 ± 0.07**	**88.66 ± 0.56**		**82.58 ± 0.24**
	5-Ensemble [Lakshminarayanan et al., 2017]	-	Predictive Entropy	Predictive Entropy	**81.80 ± 0.10**	**3.67 ± 0.11**	83.68 ± 0.33		81.12 ± 0.13
				Mutual Information			85.11 ± 0.57		81.94 ± 0.06

Table C.3: *OoD detection performance of different baselines using a DenseNet-121 architecture with the CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet dataset pairs averaged over 25 runs.* Note: SN stands for Spectral Normalization, JP stands for Jacobian Penalty. We highlight the best deterministic and best method overall in bold for each metric.

Train Dataset	**Method**	**Penalty**	Aleatoric Uncertainty	**Epistemic Uncertainty**	Accuracy (↑)	ECE (↓)	**AUROC**		
							SVHN (↑)	**CIFAR-100 (↑)**	**Tiny-ImageNet (↑)**
CIFAR-10	Softmax	-	Softmax Entropy	Softmax Entropy	95.16 ± 0.03	1.10 ± 0.04	94 ± 0.44	87.55 ± 0.11	86.99 ± 0.12
	Energy-based [Liu et al., 2020b]	-		Softmax Density			94.07 ± 0.54	86.73 ± 0.15	86.43 ± 0.16
	DUQ [van Amersfoort et al., 2020]	JP	Kernel Distance	Kernel Distance	95.02 ± 0.14	1.08 ± 0.08	94.67 ± 0.41	87.38 ± 0.21	86.72 ± 0.14
	SNGP [Liu et al., 2020a]	SN	Predictive Entropy	Predictive Entropy	94.31 ± 0.21	1.08 ± 0.10	94.48 ± 0.34	88.86 ± 0.46	88.40 ± 0.48
	DDU (ours)	SN	Softmax Entropy	GMM Density	$\mathbf{95.21 \pm 0.03}$	$\mathbf{1.05 \pm 0.03}$	$\mathbf{96.21 \pm 0.31}$	$\mathbf{90.84 \pm 0.06}$	$\mathbf{89.70 \pm 0.06}$
	5-Ensemble [Lakshminarayanan et al., 2017]	-	Predictive Entropy	Predictive Entropy	$\mathbf{96.18 \pm 0.05}$	1.07 ± 0.07	95.78 ± 0.11	90.65 ± 0.03	89.62 ± 0.06
				Mutual Information			95.75 ± 0.10	90.71 ± 0.04	89.34 ± 0.06
					Test Accuracy (↑)	**Test ECE (↓)**	**SVHN (↑)**		**Tiny-ImageNet (↑)**
CIFAR-100	Softmax	-	Softmax Entropy	Softmax Entropy	79.02 ± 0.08	4.11 ± 0.08	85.86 ± 0.42		81.10 ± 0.07
	Energy-based [Liu et al., 2020b]	-		Softmax Density			87.09 ± 0.49		80.84 ± 0.08
	SNGP [Liu et al., 2020a]	SN	Predictive Entropy	Predictive Entropy	79.15 ± 0.15	6.73 ± 0.10	85.00 ± 0.12		79.76 ± 0.15
	DDU (ours)	SN	Softmax Entropy	GMM Density	$\mathbf{79.15 \pm 0.07}$	$\mathbf{4.11 \pm 0.06}$	$\mathbf{88.44 \pm 0.55}$		$\mathbf{81.85 \pm 0.11}$
	5-Ensemble [Lakshminarayanan et al., 2017]	-	Predictive Entropy	Predictive Entropy	$\mathbf{81.01 \pm 0.13}$	4.81 ± 0.05	88.32 ± 0.61		81.45 ± 0.12
				Mutual Information			88.36 ± 0.17		81.73 ± 0.06

Table C.4: *OoD detection performance of different ablations trained on CIFAR-10 using Wide-ResNet-28-10 and VGG-16 architectures with SVHN, CIFAR-100 and Tiny-ImageNet as OoD datasets averaged over 25 runs.* Note: SN stands for Spectral Normalization. We highlight the best deterministic and best method overall in bold for each metric.

Ablations					Aleatoric Uncertainty	Epistemic Uncertainty	Test Accuracy (↑)	Test ECE (↓)	AUROC		
Architecture	Ensemble	Residual Connections	SN	GMM					SVHN (↑)	CIFAR-100 (↑)	Tiny-ImageNet (↑)
Ablations					Aleatoric Uncertainty	Epistemic Uncertainty	Test Accuracy (↑)	Test ECE (↓)	AUROC		
Architecture	Ensemble	Residual Connections	SN	GMM					SVHN (↑)	CIFAR-100 (↑)	Tiny-ImageNet (↑)
Wide-ResNet-28-10	✗	✓	✗	✗	Softmax Entropy	Softmax Entropy Softmax Density	**95.98 ± 0.02**	0.85 ± 0.02	94.44 ± 0.43 94.56 ± 0.51	89.39 ± 0.06 88.89 ± 0.07	88.42 ± 0.05 88.11 ± 0.06
				✓	Softmax Entropy	GMM Density	95.98 ± 0.02	0.85 ± 0.02	96.08 ± 0.25	90.94 ± 0.03	90.62 ± 0.05
			✓	✗	Softmax Entropy	Softmax Entropy Softmax Density	95.97 ± 0.03	0.85 ± 0.04	94.05 ± 0.26 94.31 ± 0.33	90.02 ± 0.07 89.78 ± 0.08	89.07 ± 0.06 88.96 ± 0.07
				✓	**Softmax Entropy**	**GMM Density**	95.97 ± 0.03	**0.85 ± 0.04**	**97.86 ± 0.19**	**91.34 ± 0.04**	**91.07 ± 0.05**
	✓	✓	✗	✗	Predictive Entropy	Predictive Entropy Mutual Information	**96.59 ± 0.02**	**0.76 ± 0.03**	97.73 ± 0.31 97.18 ± 0.19	**92.13 ± 0.02** 91.33 ± 0.03	90.06 ± 0.03 90.90 ± 0.03
VGG-16	✗	✓	✗	✗	Softmax Entropy	Softmax Entropy Softmax Density	93.63 ± 0.04	1.64 ± 0.03	85.76 ± 0.84 84.24 ± 1.04	82.48 ± 0.14 81.91 ± 0.17	83.07 ± 0.12 82.82 ± 0.14
				✓	Softmax Entropy	GMM Density	93.63 ± 0.04	1.64 ± 0.03	89.25 ± 0.36	86.55 ± 0.10	86.78 ± 0.09
			✓	✗	Softmax Entropy	Softmax Entropy Softmax Density	93.62 ± 0.04	1.78 ± 0.04	87.54 ± 0.41 86.28 ± 0.51	82.71 ± 0.09 82.15 ± 0.11	83.33 ± 0.08 83.07 ± 0.10
				✓	Softmax Entropy	GMM Density	93.62 ± 0.04	1.78 ± 0.04	89.62 ± 0.37	86.37 ± 0.14	86.63 ± 0.11
	✓	✓	✗	✗	Predictive Entropy	Predictive Entropy Mutual Information	94.9 ± 0.05	2.03 ± 0.03	92.80 ± 0.18 91 ± 0.22	89.01 ± 0.08 88.43 ± 0.08	87.66 ± 0.08 88.74 ± 0.05

Table C.5: *OoD detection performance of different ablations trained on CIFAR-100 using Wide-ResNet-28-10 and VGG-16 architectures with SVHN and Tiny-ImageNet as the OoD dataset averaged over 25 runs.* Note: SN stands for Spectral Normalization. We highlight the best deterministic and best method overall in bold for each metric.

Ablations					Aleatoric Uncertainty	Epistemic Uncertainty	Test Accuracy (↑)	Test ECE (↓)	AUROC	
Architecture	Ensemble	Residual Connections	SN	GMM					SVHN (↑)	Tiny-ImageNet (↑)
Wide-ResNet-28-10	✗	✓	✗	✗	Softmax Entropy	Softmax Entropy Softmax Density	80.26 ± 0.06	4.62 ± 0.06	77.42 ± 0.57 78.00 ± 0.63	81.53 ± 0.05 81.33 ± 0.06
				✓	Softmax Entropy	GMM Density	80.26 ± 0.06	4.62 ± 0.06	87.54 ± 0.61	78.13 ± 0.08
			✓	✗	Softmax Entropy	Softmax Entropy Softmax Density	80.98 ± 0.06	4.10 ± 0.08	85.37 ± 0.36 86.41 ± 0.38	82.57 ± 0.03 82.49 ± 0.04
				✓	**Softmax Entropy**	**GMM Density**	$\mathbf{80.98 \pm 0.06}$	$\mathbf{4.10 \pm 0.08}$	$\mathbf{87.53 \pm 0.62}$	$\mathbf{83.13 \pm 0.06}$
	✓	✓	✗	✗	Predictive Entropy	Predictive Entropy Mutual Information	$\mathbf{82.79 \pm 0.10}$	$\mathbf{3.32 \pm 0.09}$	79.54 ± 0.91 77.00 ± 1.54	82.95 ± 0.09 82.82 ± 0.04
VGG-16	✗	✓	✗	✗	Softmax Entropy	Softmax Entropy Softmax Density	73.48 ± 0.05	4.46 ± 0.05	76.73 ± 0.72 77.70 ± 0.86	76.43 ± 0.05 74.68 ± 0.07
				✓	Softmax Entropy	GMM Density	73.48 ± 0.05	4.46 ± 0.05	75.65 ± 0.95	74.32 ± 1.73
			✓	✗	Softmax Entropy	Softmax Entropy Softmax Density	73.58 ± 0.06	4.32 ± 0.06	77.21 ± 0.77 77.76 ± 0.90	76.59 ± 0.06 74.86 ± 0.08
				✓	Softmax Entropy	GMM Density	73.58 ± 0.06	4.32 ± 0.06	75.99 ± 1.23	74.06 ± 1.67
	✓	✓	✗	✗	Predictive Entropy	Predictive Entropy Mutual Information	77.84 ± 0.11	5.32 ± 0.10	79.62 ± 0.73 72.07 ± 0.48	78.66 ± 06 76.27 ± 0.05

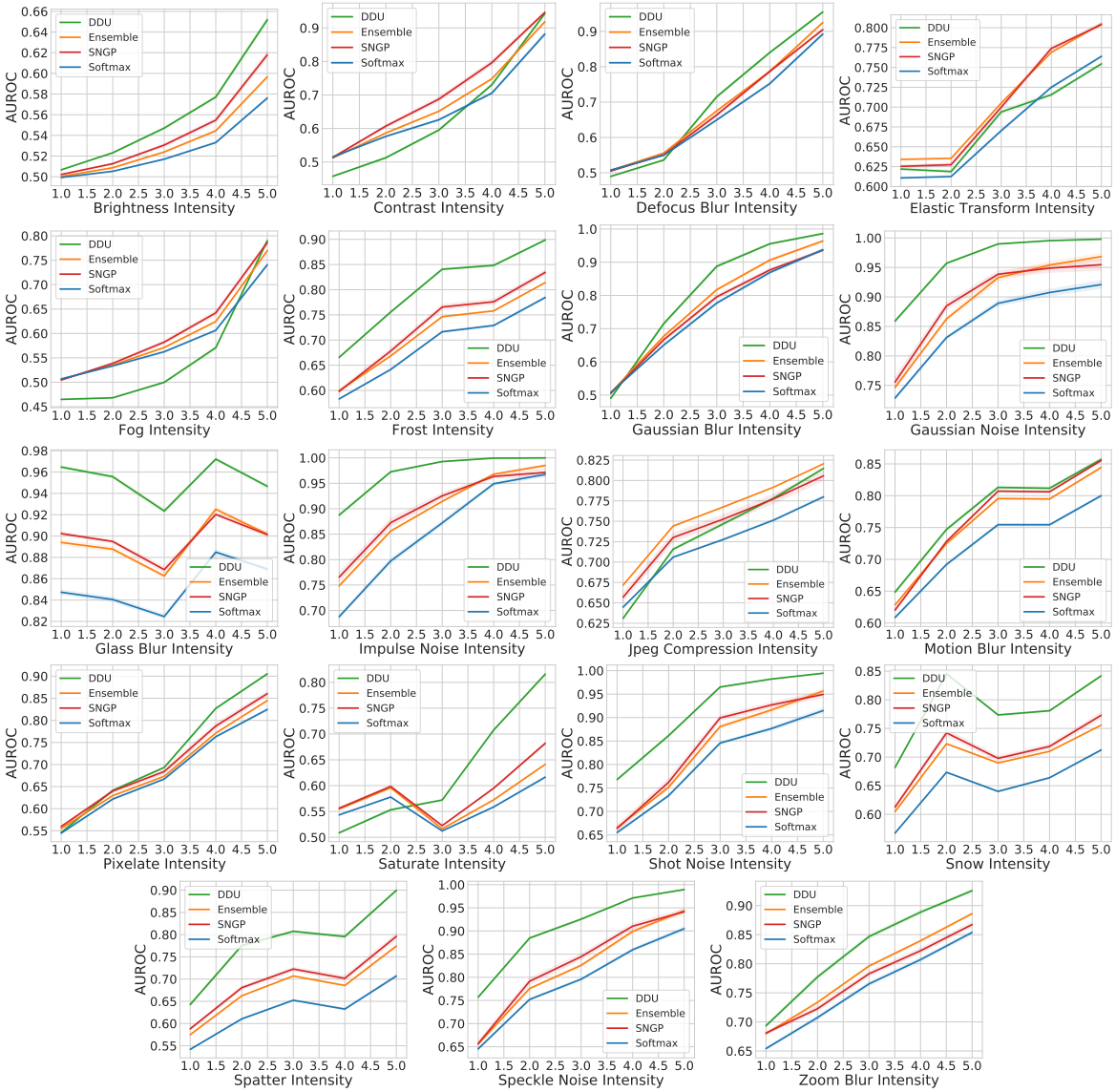


Figure C.2: AUROC vs corruption intensity for all corruption types in CIFAR-10-C with Wide-ResNet-28-10 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density.

2. **Effect of sensitivity alone:** Since residual connections make a model sensitive to changes in the input space by lower bounding its Lipschitz constant, we also want to see how a network performs with just the sensitivity constraint alone. To observe this, we train a Wide-ResNet-28-10 without spectral normalization (i.e. no explicit upper bound on the Lipschitz constant of the model).
3. **Metrics for aleatoric and epistemic uncertainty:** With the above combinations, we try to observe how different metrics for aleatoric and epistemic uncertainty perform. To quantify aleatoric uncertainty, we use the softmax entropy of the model. On the other hand, to quantify the epistemic uncertainty, we use **i)** the softmax entropy, **ii)** the softmax density [Liu et al., 2020b] or **iii)** the GMM feature density (as described in §3.4).

For the purposes of comparison, we also present scores obtained by a 5-Ensemble of the respective architectures (i.e. Wide-ResNet-28-10 and VGG-16) in Table C.4 for

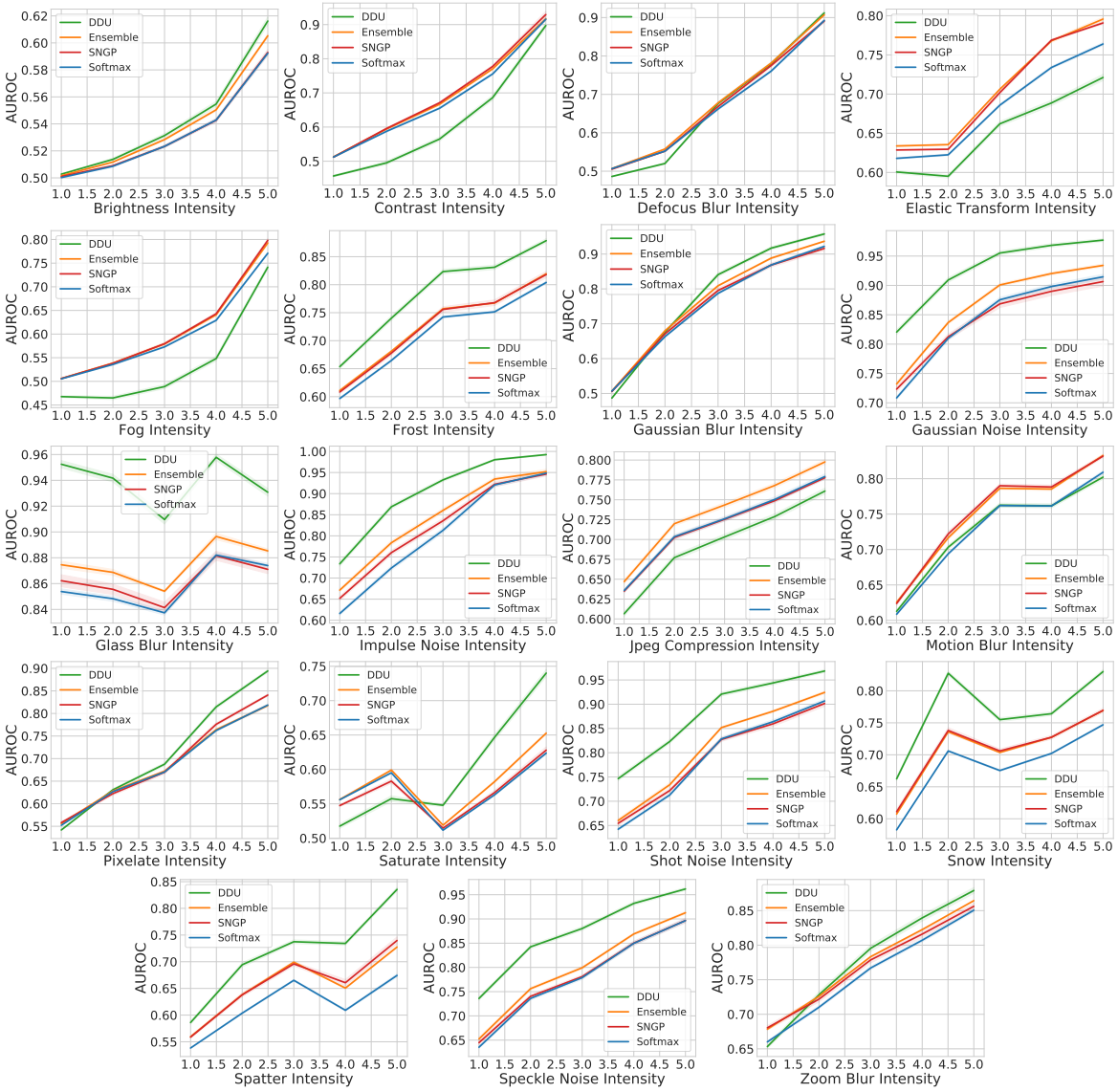


Figure C.3: AUROC vs corruption intensity for all corruption types in CIFAR-10-C with ResNet-50 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density.

CIFAR-10 vs SVHN/CIFAR-100 and in Table C.5 for CIFAR-100 vs SVHN. Based on these results, we can make the following observations (in addition to the ones we make in §3.5.3):

Inductive biases are important for feature density. From the AUROC scores in Table C.4, we can see that using the feature density of a GMM in VGG-16 without the proposed inductive biases yields significantly lower AUROC scores as compared to Wide-ResNet-28-10 with inductive biases. In fact, in none of the datasets is the feature density of a VGG able to outperform its corresponding ensemble. This provides yet more evidence (in addition to Figure 3.2) to show that the GMM feature density alone cannot estimate epistemic uncertainty in a model that suffers from feature collapse. We need sensitivity and smoothness conditions (see §3.2) on the feature space of the model to obtain feature densities that capture epistemic uncertainty.

Sensitivity creates a bigger difference than smoothness. We note that the

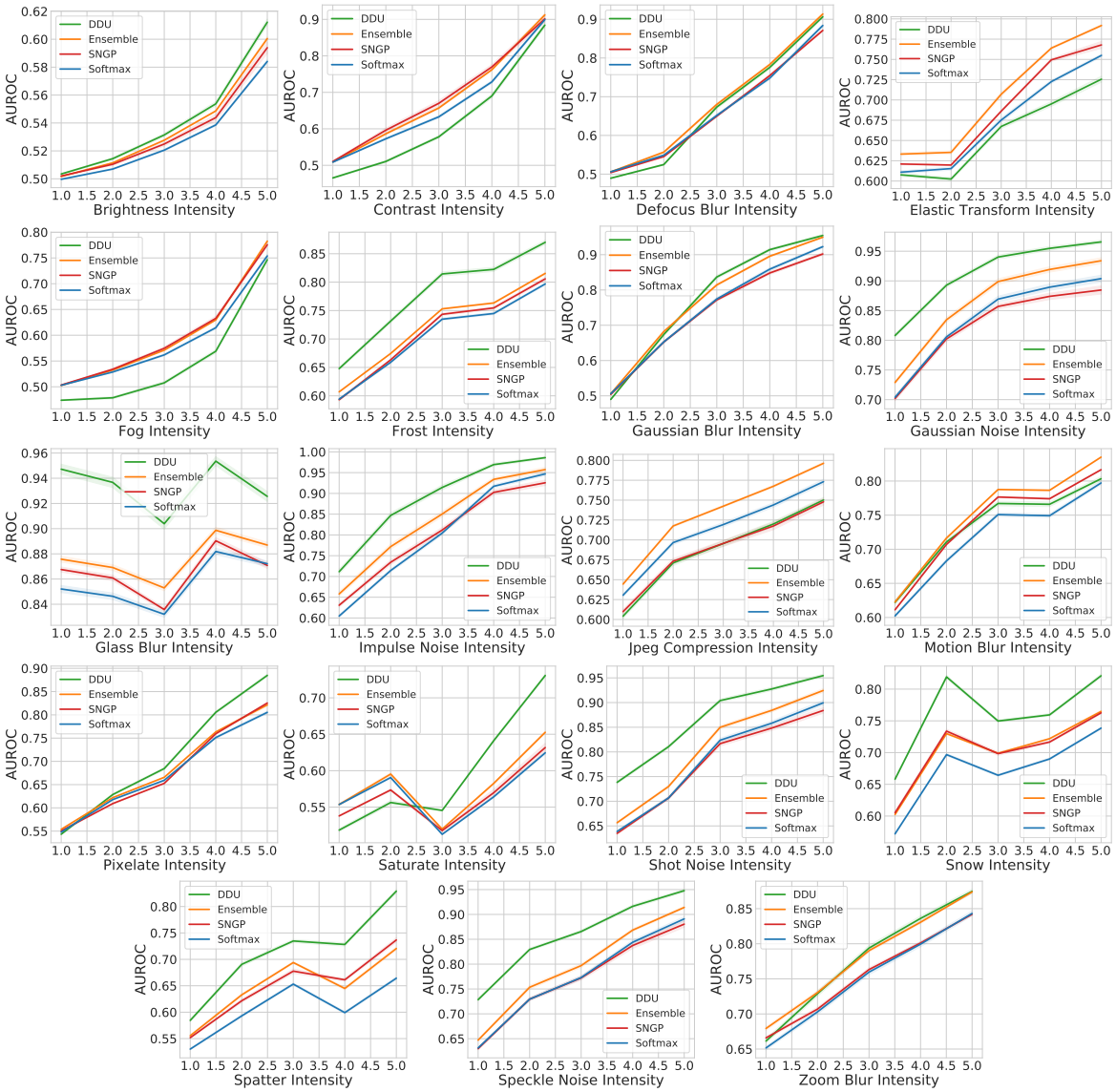


Figure C.4: AUROC vs corruption intensity for all corruption types in CIFAR-10-C with ResNet-110 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density.

difference between AUROC obtained from feature density between Wide-ResNet-28-10 models with and without spectral normalization is minimal. Although Wide-ResNet-28-10 with spectral normalization (i.e. smoothness constraints) still outperforms its counterpart without spectral normalization, the small difference between the AUROC scores indicates that it might be the residual connections (i.e. sensitivity constraints) that make the model detect OoD samples better. This observation is also intuitive as a sensitive feature extractor should map OoD samples farther from iD ones.

DDU as a simple baseline. In DDU, we use the softmax output of a model to get aleatoric uncertainty. We use the GMM's feature-density to estimate the epistemic uncertainty. Hence, DDU does not suffer from miscalibration as the softmax outputs can be calibrated using post-hoc methods like temperature scaling. At the same time, the feature-densities of the model are not affected by temperature scaling and capture epistemic uncertainty well.

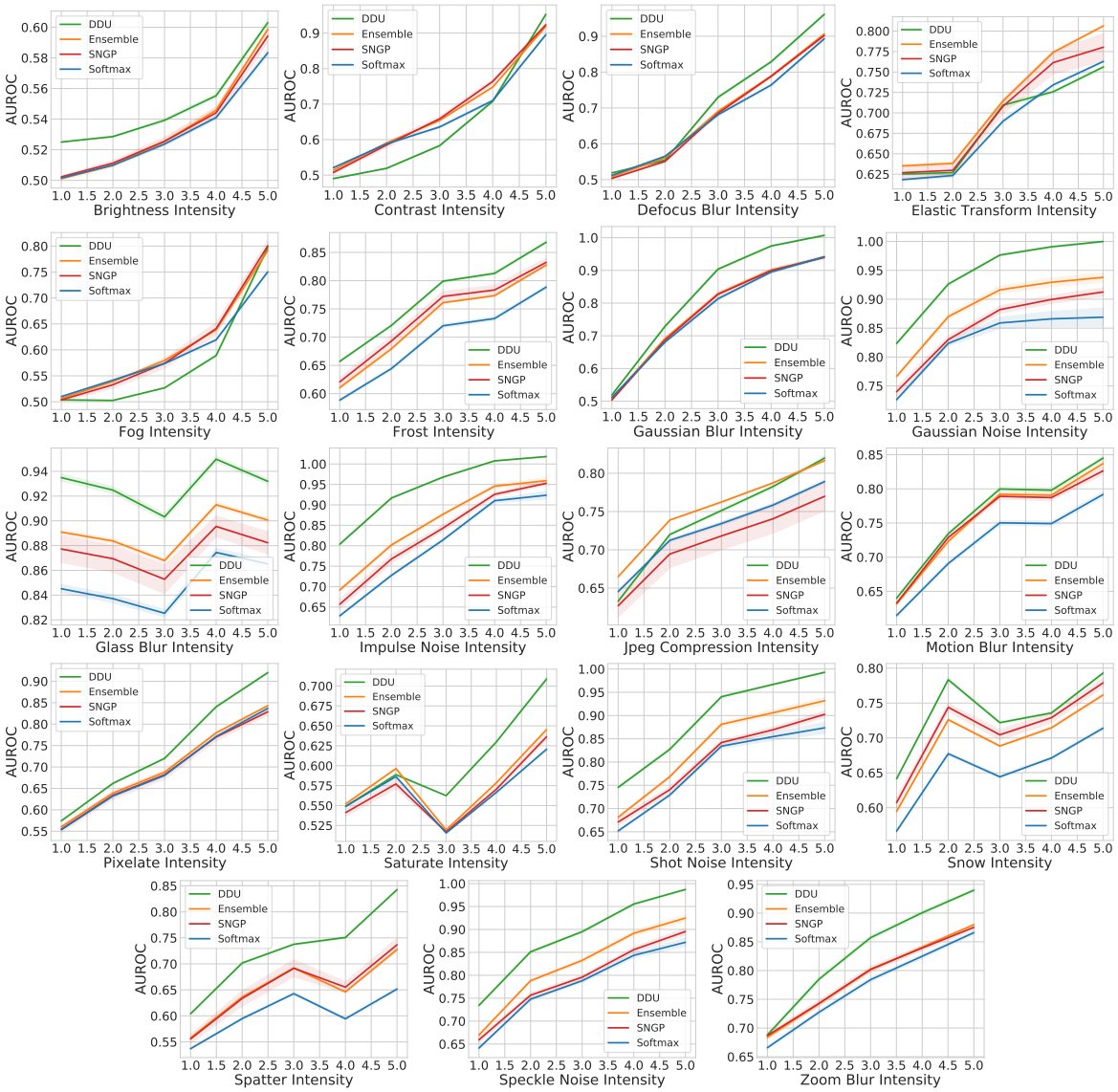


Figure C.5: AUROC vs corruption intensity for all corruption types in CIFAR-10-C with DenseNet-121 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density.

C.3 Additional Ablations & Toy Experiments

Here, we provide details for the toy experiments mentioned in the main part of the thesis which are visualized in Figure 3.2, Figure 3.3 and Figure 3.13.

C.3.1 QUBIQ Challenge

In this section, we evaluate DDU's performance on the real-world QUBIQ challenge related to biomedical imaging. QUBIQ has a total of 7 binary segmentation tasks in 4 biomedical imaging datasets with multiple annotations per image. The task is to predict the distribution of source labels with a mask of values between 0 and 1. For evaluation, the annotations are averaged to provide a continuous ground-truth. The prediction mask and continuous ground-truth are binarized by thresholding between $[0, 1]$ and a Dice score is computed between the resulting binary masks. The average

Table C.6: Dice scores for the QUBIQ 2021 challenge.

Method	Softmax	Energy	3-Ensemble PE	DDU
Dice Score (↑)	78.4 ± 1.31	77.31 ± 1.5	82.25 ± 0.83	$\mathbf{82.63 \pm 1.08}$

Dataset	Metric	Softmax & Energy	DUQ	SNGP	DDU	5-Ensemble
CIFAR-10	ECE	$\mathbf{0.85 \pm 0.02}$	1.55 ± 0.08	1.8 ± 0.1	$\mathbf{0.85 \pm 0.04}$	$\mathbf{0.76 \pm 0.03}$
	TACE	0.63 ± 0.01	0.84 ± 0.03	0.9 ± 0.04	$\mathbf{0.61 \pm 0.01}$	$\mathbf{0.48 \pm 0.01}$
	NLL	0.18 ± 0.06	0.23 ± 0.07	0.27 ± 0.08	$\mathbf{0.16 \pm 0.06}$	$\mathbf{0.11 \pm 0.02}$
CIFAR-100	ECE	4.62 ± 0.06	-	4.33 ± 0.01	$\mathbf{4.1 \pm 0.08}$	$\mathbf{3.32 \pm 0.09}$
	TACE	1.31 ± 0.02	-	1.23 ± 0.04	$\mathbf{1.06 \pm 0.03}$	$\mathbf{0.58 \pm 0.03}$
	NLL	1.17 ± 0.13	-	0.92 ± 0.16	$\mathbf{0.86 \pm 0.14}$	$\mathbf{0.73 \pm 0.09}$

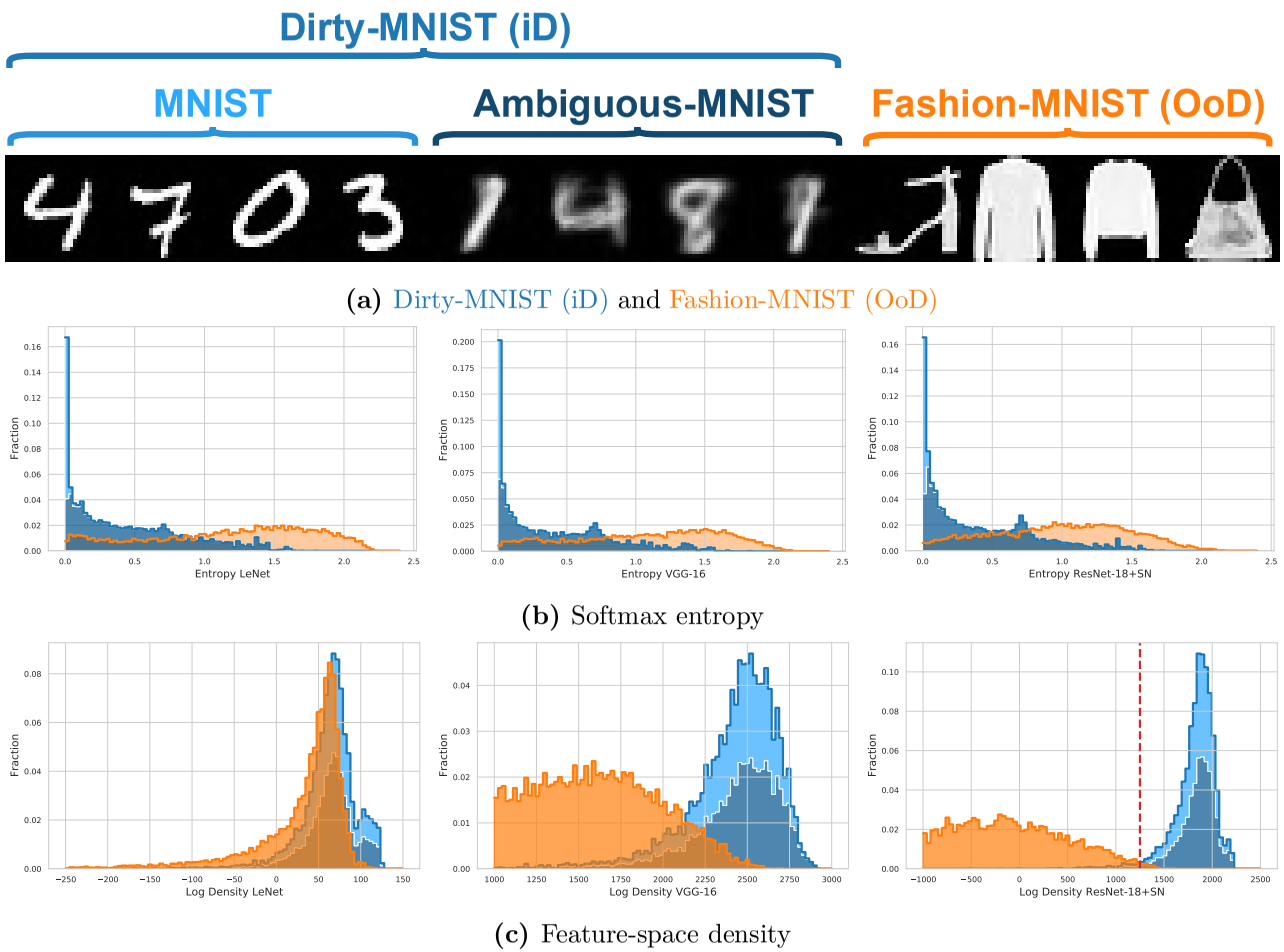
Table C.7: Calibration error scores ECE%, TACE% and NLL for WRN-28-10.

dice score across thresholds, images and tasks is reported. Note that in the continuous ground-truth, 0.5 indicates maximum uncertainty and values above or below indicate lower uncertainty. Thus, for our comparison, we scale all uncertainty values to the range $u \in [0, 0.5]$ and use $p + u$ if $p = 0$ and $p - u$ if $p = 1$, where p is the binary prediction. We use a UNet model with a ResNet encoder and report the Dice scores averaged over 5 runs in Table C.6. Even on this real-world dataset, DDU performs as well as ensembles and outperforms Softmax and Energy baselines.

C.3.2 Additional Calibration Metrics

In Table C.7, in addition to ECE%, we provide additional calibration error scores: temperature scaled Thresholded Adaptive Calibration Error (TACE) % and Negative Log Likelihood (NLL) for Wide-ResNet-28-10 trained on CIFAR-10/100—the main results for this can be found in Table 3.5. Here, we see that the results for TACE and NLL are consistent with what we see for ECE. Ensembles produce the most calibrated models and among deterministic baselines, DDU is the best calibrated.

C.4 Big Figure 1



(a) Dirty-MNIST (iD) and Fashion-MNIST (OoD)

(b) Softmax entropy

(c) Feature-space density

Figure C.6: *Disentangling aleatoric and epistemic uncertainty on Dirty-MNIST (iD) and Fashion-MNIST (OoD) **(a)** requires using* softmax entropy *__(b)__ and* feature-space density (GMM) *__(c)__ with appropriate inductive biases (*ResNet-18+SN *vs* LeNet *&* VGG-16 *without them). Enlarged version.* **(b):** Softmax entropy captures aleatoric uncertainty for iD data (Dirty-MNIST), thereby separating unambiguous MNIST samples and Ambiguous-MNIST samples. However, iD and OoD are confounded: softmax entropy has arbitrary values for OoD, indistinguishable from iD. **(c):** With appropriate inductive biases (DDU with ResNet-18+SN), iD and OoD densities do not overlap, capturing epistemic uncertainty. However, without appropriate inductive biases (LeNet & VGG-16), feature density suffers from *feature collapse*: iD and OoD densities overlap. Generally, feature-space density confounds unambiguous and ambiguous iD samples as their densities overlap. **Note:** Unambiguous MNIST samples and Ambiguous-MNIST samples are shown as stacked histograms with the total fractions adding up to 1 for Dirty-MNIST.

D

Diverse Batch Acquisition for Bayesian Active Learning

D.1 Proof of Submodularity

Nemhauser et al. [1978] show that if a function is submodular, then a greedy algorithm like algorithm 2 is $1 - 1/e$-approximate. Here, we show that $a_{\texttt{BatchBALD}}$ is submodular.

We will show that $a_{\texttt{BatchBALD}}$ satisfies the following equivalent definition of submodularity:

Definition D.1. A function f defined on subsets of Ω is called *submodular* if for every set $A \subset \Omega$ and two non-identical points $X, Y \in \Omega \setminus A$:

$$f(A \cup \{X\}) + f(A \cup \{Y\}) \geq f(A \cup \{X, Y\}) + f(A) \tag{D.1}$$

Submodularity expresses that there are "diminishing returns" for adding additional points to f.

Lemma D.1. $a_{\texttt{BatchBALD}}(A, \mathrm{p}(\boldsymbol{\omega})) := \mathrm{I}[A; \boldsymbol{\Omega}]$ *is submodular for* $A \subset \mathcal{D}^{\text{pool}}$.

Proof. Let $X, Y \in \mathcal{D}^{\text{pool}}, X \neq Y$. We start by substituting the definition of $a_{\texttt{BatchBALD}}$ into (D.1) and subtracting $\mathrm{I}[A; \boldsymbol{\Omega}]$ twice on both sides, using that $\mathrm{I}[A \cup B; \boldsymbol{\Omega}] - \mathrm{I}[B; \boldsymbol{\Omega}] = \mathrm{I}[A; \boldsymbol{\Omega} \mid B]$:

$$\mathrm{I}[A \cup \{Y\}; \boldsymbol{\Omega}] + \mathrm{I}[A \cup \{X\}; \boldsymbol{\Omega}] \geq \mathrm{I}[A \cup \{X, Y\}; \boldsymbol{\Omega}] + \mathrm{I}[A; \boldsymbol{\Omega}] \tag{D.2}$$

$$\Leftrightarrow\ \mathrm{I}[Y; \boldsymbol{\Omega} \mid A] + \mathrm{I}[X; \boldsymbol{\Omega} \mid A] \geq \mathrm{I}[X, y; \boldsymbol{\Omega} \mid A]. \tag{D.3}$$

We rewrite the left-hand side using the definition of the mutual information $\mathrm{I}[A; B] = \mathrm{H}[A] - \mathrm{H}[A \mid B]$ and reorder:

$$\mathrm{I}[y; \boldsymbol{\Omega} \mid A] + \mathrm{I}[x; \boldsymbol{\Omega} \mid A] \tag{D.4}$$

$$= \underbrace{\mathrm{H}[X \mid A] + \mathrm{H}[X \mid A]}_{\geq \mathrm{H}[X, Y \mid A]} - \underbrace{(\mathrm{H}[X \mid A, \boldsymbol{\Omega}] + \mathrm{H}[Y \mid A, \boldsymbol{\Omega}])}_{= \mathrm{H}[X, Y \mid A, \boldsymbol{\Omega}]} \tag{D.5}$$

$$\geq \mathrm{H}[X, Y \mid A] - \mathrm{H}[X, Y \mid A, \boldsymbol{\Omega}] \tag{D.6}$$

$$= \mathrm{I}[X, Y; \boldsymbol{\Omega} \mid A], \tag{D.7}$$

where we have used that entropies are subadditive in general and additive given $X \perp\!\!\!\perp Y \mid \boldsymbol{\Omega}$. □

Following Nemhauser et al. [1978], we can conclude that algorithm 2 is $1 - 1/e$-approximate.

D.2 BALD as an Upper-Bound of BatchBALD

In the following section, we show that BALD approximates BatchBALD. The BALD score is an upper bound of the BatchBALD score for any candidate batch.

Using the subadditivity of information entropy and the independence of the y_i given $\boldsymbol{\omega}$, we show that BALD is an approximation of BatchBALD and is always an upper bound on the respective BatchBALD score:

$$a_{\texttt{BatchBALD}}\left(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_b\}, \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})\right) \tag{D.8}$$

$$= \mathrm{I}[Y^{\text{acq}}_{1..\mathtt{K}}; \boldsymbol{\Omega} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}] \tag{D.9}$$

$$= \mathrm{H}[Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y^{\text{acq}}_{1..\mathtt{K}} \mid \boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}] \tag{D.10}$$

$$\leq \sum_{i=1}^{\mathtt{K}} \mathrm{H}[Y_i \mid \boldsymbol{x}_i, \mathcal{D}^{\text{train}}] - \sum_{i=1}^{\mathtt{K}} \mathrm{H}[Y_i \mid \boldsymbol{x}_i, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}] \tag{D.11}$$

$$= \sum_{i=1}^{\mathtt{K}} \mathrm{I}[Y_i; \boldsymbol{\Omega} \mid \boldsymbol{x}_i, \mathcal{D}^{\text{train}}] = a_{\texttt{BALD}}\left(\{\boldsymbol{x}^{\text{acq}}_{1..\mathtt{K}}\}, \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})\right) \tag{D.12}$$

Relevance for the active training loop. We see that the active training loop as a whole is computing a greedy $1 - 1/e$-approximation of the mutual information of all acquired data points over all acquisitions with the model parameters.

D.3 Sampling of Configurations

We are using the same notation as in section 4.1.2. We factor $\mathrm{p}(y_{1..n} \mid \boldsymbol{\omega})$ to avoid recomputations and rewrite $\mathrm{H}[Y_{1..n}]$ as:

$$\mathrm{H}[Y_{1..n}] = \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})} \mathbb{E}_{\mathrm{p}(y_{1..n}|\boldsymbol{\omega})}[-\log \mathrm{p}(y_{1..n})] \tag{D.13}$$

$$= \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})} \mathbb{E}_{\mathrm{p}(y_{1:n-1}|\boldsymbol{\omega})\,\mathrm{p}(y_n|\boldsymbol{\omega})}[-\log \mathrm{p}(y_{1..n})] \tag{D.14}$$

$$= \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})} \mathbb{E}_{\mathrm{p}(y_{1:n-1}|\boldsymbol{\omega})} \mathbb{E}_{\mathrm{p}(y_n|\boldsymbol{\omega})}[-\log \mathrm{p}(y_{1..n})] \tag{D.15}$$

To be flexible in the way we sample $y_{1:n-1}$, we perform importance sampling of $\mathrm{p}(y_{1:n-1} \mid \boldsymbol{\omega})$ using $\mathrm{p}(y_{1:n-1})$, and, assuming we also have m samples $\hat{y}_{1:n-1}$ from $\mathrm{p}(y_{1:n-1})$, we can approximate:

$$\mathrm{H}[Y_{1..n}] = \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})} \mathbb{E}_{\mathrm{p}(y_{1:n-1})}\left[\frac{\mathrm{p}(y_{1:n-1} \mid \boldsymbol{\omega})}{\mathrm{p}(y_{1:n-1})} \mathbb{E}_{\mathrm{p}(y_n|\boldsymbol{\omega})}[-\log \mathrm{p}(y_{1:n})]\right] \tag{D.16}$$

$$= \mathbb{E}_{\mathrm{p}(y_{1:n-1})} \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})} \mathbb{E}_{\mathrm{p}(y_n|\boldsymbol{\omega})}\left[-\frac{\mathrm{p}(y_{1:n-1} \mid \boldsymbol{\omega})}{\mathrm{p}(y_{1:n-1})} \log \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}[\mathrm{p}(y_{1:n-1} \mid \boldsymbol{\omega})\,\mathrm{p}(y_{1:n} \mid \boldsymbol{\omega})]\right] \tag{D.17}$$

$$\approx -\frac{1}{m} \sum_{\hat{y}_{1:n-1}}^{m} \sum_{\hat{y}_n} \frac{\frac{1}{k}\sum_{\hat{\boldsymbol{\omega}}_j} \mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{\omega}}_j)\,\mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{\omega}}_j)}{\mathrm{p}(\hat{y}_{1:n-1})} \log\left(\frac{1}{k}\sum_{\hat{\boldsymbol{\omega}}_j} \mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{\omega}}_j)\,\mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{\omega}}_j)\right) \tag{D.18}$$

$$= -\frac{1}{m} \sum_{\hat{y}_{1:n-1}}^{m} \sum_{\hat{y}_n} \frac{\left(\hat{P}_{1:n-1}\hat{P}_n^T\right)_{\hat{y}_{1:n-1},\hat{y}_n}}{\left(\hat{P}_{1:n-1}\mathbb{1}_{k,1}\right)_{\hat{y}_{1:n-1}}} \log\left(\frac{1}{k}\left(\hat{P}_{1:n-1}\hat{P}_n^T\right)_{\hat{y}_{1:n-1},\hat{y}_n}\right), \tag{D.19}$$

where we store $\mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{\omega}}_j)$ in a matrix $\hat{P}_{1:n-1}$ of shape $m \times k$ and $\mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{\omega}}_j)$ in a matrix $\hat{P}_n$ of shape $\mathtt{C} \times k$ and $\mathbb{1}_{k,1}$ is a $k \times 1$ matrix of 1s. Equation (D.19) allows us to cache $\hat{P}_{1:n-1}$ inside the inner loop of algorithm 2 and use batch matrix multiplication for efficient computation.

D.4 Ablation Study on Repeated-MNIST

To better understand the effect of redundant data points on BALD and BatchBALD, we run the RMNIST experiment with an increasing number of repetitions. The results can be seen in figure D.1. We use the same setup as in section 4.2.1. BatchBALD performs the same on all repetition numbers (100 data points till 90%). BALD achieves 90% accuracy at 120 data points (0 repetitions), 160 data points (1 repetition), 280 data points (2 repetitions), 300 data points (4 repetitions). This shows that BALD and BatchBALD behave as expected.

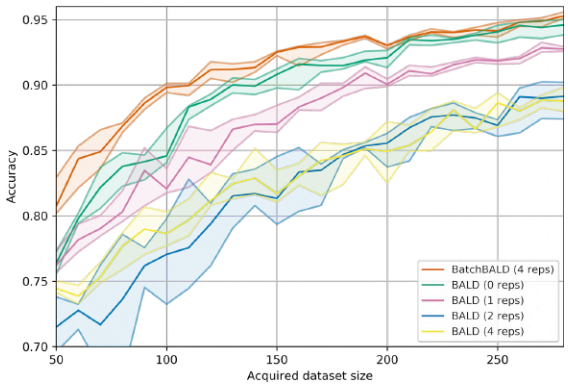


Figure D.1: *Performance of* BALD *on* Repeated MNIST *for increasing amount of repetitions.* We see that BALD performs worse as the number of repetitions is increased, while BatchBALD outperforms BALD with zero repetitions.

D.5 Additional Results for Repeated-MNIST

We show that BatchBALD also outperforms Var Ratios [Freeman, 1965] and Mean STD [Kendall et al., 2017].

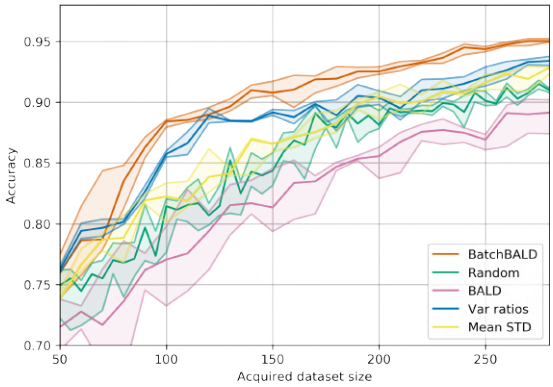


Figure D.2: *Performance on* Repeated MNIST. BALD, BatchBALD, Var Ratios, Mean STD and random acquisition with acquisition size 10 and 10 MC dropout samples.

D.6 Example Visualisation of EMNIST

Figure D.3: *Examples of all 47 classes of EMNIST*

D.7 Entropy and Per-Class Acquisitions (including Random Acquisition)

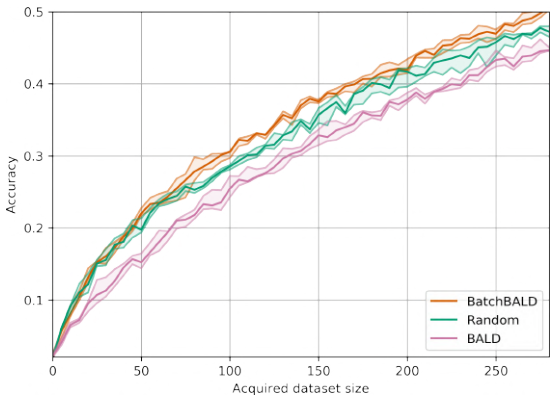


Figure D.4: *Performance on* EMNIST. BatchBALD consistently outperforms both random acquisition and BALD while BALD is unable to beat random acquisition.

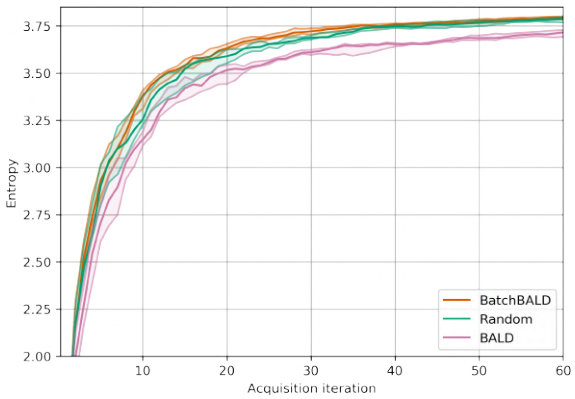


Figure D.5: *Entropy of acquired class labels over acquisition steps on* EMNIST. BatchBALD steadily acquires a more diverse set of data points than BALD.

E

Stochastic Batch Acquisition for Deep Active Learning

E.1 Proof of Proposition 5.1

First, we remind the reader that a random variable G is Gumble distributed $G \sim \text{Gumbel}(\mu; \beta)$ when its cumulative distribution function follows $\text{p}(G \leq g) = \exp(-\exp(-\frac{g-\mu}{\beta}))$.

Furthermore, the Gumbel distribution is closed under translation and positive scaling:

Lemma E.1. *Let $G \sim$ Gumbel$(\mu; \beta)$ be a Gumbel distributed random variable, then:*

$$\alpha G + d \sim \textit{Gumbel}(d + \alpha\mu; \alpha\beta). \tag{E.1}$$

Proof. We have $\text{p}(\alpha G + d \leq x) = \text{p}(G \leq \frac{x-d}{\alpha})$. Thus, we have:

$$\text{p}(\alpha G + d \leq x) = \exp(-\exp(-\frac{\frac{x-d}{\alpha} - \mu}{\beta})) \tag{E.2}$$

$$= \exp(-\exp(-\frac{x - (d + \alpha\mu)}{\alpha\beta})) \tag{E.3}$$

$$\Leftrightarrow \alpha G + d \sim \text{Gumbel}(d + \alpha\mu; \alpha\beta). \tag{E.4}$$

□

We can then easily prove Proposition 5.1 using Theorem 1 from Kool et al. [2019], which we present it here slightly reformulated to fit our notation:

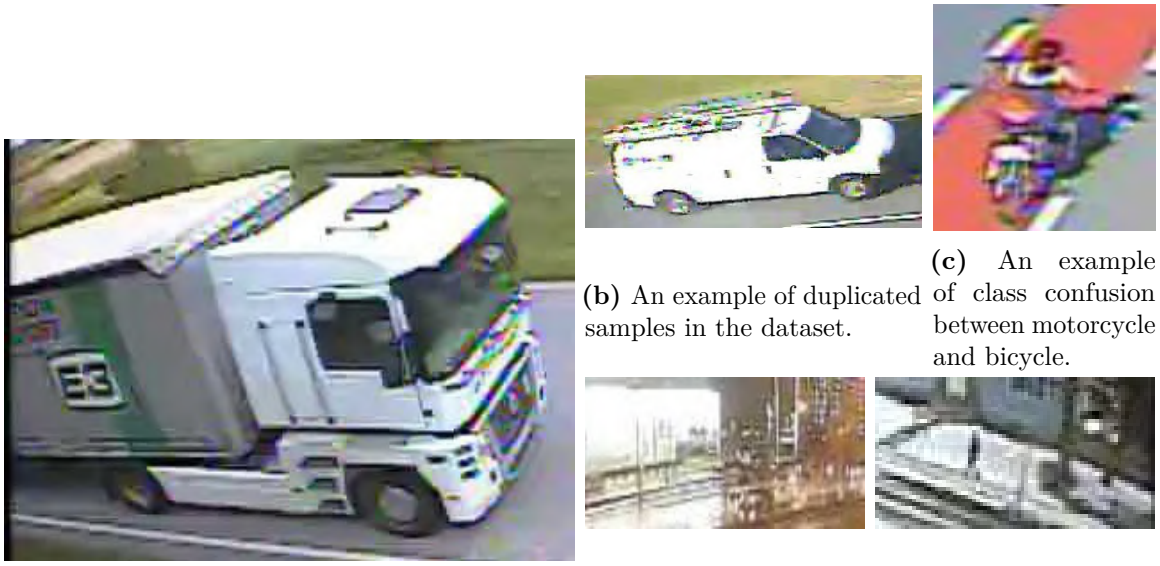
Lemma E.2. *For $k \leq n$, let $I_1^*, \ldots, I_k^* = \arg\text{top}_k\{s_i + \epsilon_i\}_i$ with $\epsilon_i \sim$ Gumbel$(0; 1)$, i.i.d.. Then $I_1^*, \ldots, I_k^*$ is an (ordered) sample without replacement from the categorical distribution*

$$\text{Categorical}\left(\frac{\exp s_i}{\sum_{j \in n} \exp s_j}, i \in \{1, \ldots, n\}\right), \tag{E.5}$$

e.g. for a realization $i_1^*, \ldots, i_k^*$ it holds that

$$P\left(I_1^* = i_1^*, \ldots, I_k^* = i_k^*\right) = \prod_{j=1}^{k} \frac{\exp s_{i_j^*}}{\sum_{\ell \in N_j^*} \exp s_\ell} \tag{E.6}$$

where $N_j^* = N \setminus \left\{i_1^*, \ldots, i_{j-1}^*\right\}$ is the domain (without replacement) for the j-th sampled element.



(a) A good example in MIOTCD dataset.

(b) An example of duplicated samples in the dataset.

(c) An example of class confusion between motorcycle and bicycle.

(d) An example of heavy compression artifact.

(e) An example of low resolution samples.

Figure E.1: *MIO-TCD Dataset* is designed to include common artifacts from production data. The size and quality of the images vary greatly between crops; from high-quality cameras on sunny days to low-quality cameras at night. (a) shows an example of clean samples that can be clearly assigned to a class. (b)(c)(d) and (e) show the different categories of noise. (b) shows an example of many near-duplicates that exist in the dataset. (c) is a good example where the assigned class is subject to interpretation (d) is a sample with heavy compression artifacts and (e) is an example of samples with low resolution which again is considered a hard example to learn for the model.

Now, it is easy to prove the proposition:

Proposition 5.1. *For scores s_i, $i \in \{1, \dots, n\}$, and $k \leq n$ and $\beta > 0$, if we draw $\epsilon_i \sim$ Gumbel$(0; \beta^{-1})$ independently, then* $\arg\operatorname{top}_k\{s_i + \epsilon_i\}_i$ *is an (ordered) sample without replacement from the categorical distribution* $\text{Categorical}(\exp(\beta\, s_i)/\sum_j \exp(\beta\, s_j), i \in \{1, \dots, n\})$.

Proof. As $\epsilon_i \sim \text{Gumbel}(0; \beta^{-1})$, define $\epsilon'_i \triangleq \beta\epsilon_i \sim \text{Gumbel}(0; 1)$. Further, let $s'_i \triangleq \beta s_i$. Applying Lemma E.2 on s'_i and ϵ'_i, $\arg\operatorname{top}_k\{s'_i + \epsilon'_i\}_i$ yields (ordered) samples without replacement from the categorical distribution $\text{Categorical}(\frac{\exp(\beta\, s_i)}{\sum_j \exp(\beta\, s_j)}, i \in \{1, \dots, n\})$. However, multiplication by β does not change the resulting indices of $\arg\operatorname{top}_k$:

$$\arg\operatorname{top}_k\{s'_i + \epsilon'_i\}_i = \arg\operatorname{top}_k\{s_i + \epsilon_i\}_i, \tag{E.7}$$

concluding the proof. □

E.2 Empirical Validation

E.2.1 Experimental Setup & Compute

Full code for all experiments will be available at anonymized_github_repo.

Frameworks. We use PyTorch. Repeated-MNIST and EMNIST experiments use PyTorch Ignite. Synbols and MIO-TCD experiments use the BaaL library https://github.com/baal-org/baal [Atighehchian et al., 2020]. Predictive parity is calculated using FairLearn [Bird et al., 2020]. The CausalBALD experiments use https://github.com/anndvision/causal-bald [Jesson et al., 2021].

Compute. Results shown in Table 5.1 were run inside Docker containers with 8 CPUs (2.2Ghz) and 32 Gb of RAM. Other experiments were run on similar machines with Titan RTX GPUs. The Repeated-MNIST and EMNIST experiments take about 5000 GPU hours. The MIO, Synbols and CLINC-150 experiments take about 19000 GPU hours. The CausalBALD experiments take about 1000 GPU hours.

Dataset Licenses. Repeated-MNIST is based on MNIST which is made available under the terms of the Creative Commons Attribution-Share Alike 3.0 license. The EMNIST dataset is made available as CC0 1.0 Universal Public Domain Dedication. Synbols is a dataset generator. MIO-TCD is made available under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. CLINC-150 is made available under the terms of Creative Commons Attribution 3.0 Unported License.

E.2.1.1 Runtime Measurements

The synthetic dataset used for benchmarking has 4,096 features, 10 classes, and 10,000 pool points. VGG-16 models [Simonyan and Zisserman, 2015] were used to sample predictions and latent embeddings.

E.2.1.2 Repeated-MNIST

The Repeated-MNIST dataset (introduced in §4) with duplicated examples from MNIST with isotropic Gaussian noise added to the input images (standard deviation 0.1).

We use the same setup as in §4: a LeNet-5-like architecture with ReLU activations instead of tanh and added dropout. The model obtains 99% test accuracy when trained on the full MNIST dataset. Specifically, the model is made up of two blocks of a convolution, dropout, max-pooling, ReLU with 32 and 64 channels and 5x5 kernel size, respectively. As classifier head, a two-layer MLP with 128 hidden units (and 10 output units) is used that includes dropout between the layers. We use a dropout probability of 0.5 everywhere. The model is trained with early stopping using the Adam optimizer and a learning rate of 0.001. We sample predictions using 100 MC-Dropout samples for BALD. Weights are reinitialized after each acquisition step.

E.2.1.3 EMNIST

We follow the setup from §4 with 20 MC dropout samples. We use a similar model as for Repeated-MNIST but with three blocks instead of two. Specifically, we use 32, 64, and 128 channels and 3x3 kernel size. This is followed by a 2x2 max pooling layer before the classifier head. The classifier head is a two-layer MLP but with 512 hidden units instead of 128. Again, we use dropout probability 0.5 everywhere.

E.2.1.4 Synbols & MIO-TCD

The full list of hyperparameters for the Synbols and MIO-TCD experiments is presented in Table E.1. Our experiments are built using the BaaL library [Atighehchian et al., 2020]. We compute the predictive parity using FairLearn [Bird et al., 2020]. We use

Table E.1: Hyperparameters used in Section 5.4 and E.2.5

Hyperparameter	Value
Learning Rate	0.001
Optimizer	SGD
Weight Decay	0
Momentum	0.9
Loss Function	Cross-Entropy
Training Duration	10
Batch Size	32
Dropout p	0.5
MC Iterations	20
Query Size	100
Initial Set	500

VGG-16 model [Simonyan and Zisserman, 2015] trained for 10 epochs using Monte Carlo dropout for acquisition [Gal et al., 2017] with 20 dropout samples.

In Figure E.1, we show a set of images with common problems that can be found in MIO-TCD.

E.2.1.5 CLINC-150

We fine-tune a pretrained DistilBERT model from HuggingFace [Dosovitskiy et al., 2020] on CLINC-150 for 5 epochs with Adam as optimizer. Estimating epistemic uncertainty in transformer models is an open research question, and hence, we do not report results using BALD and focus on entropy instead.

E.2.1.6 CausalBALD

Using the Neyman-Rubin framework [Neyman, 1923; Rubin, 1974; Sekhon, 2008], the CATE is formulated in terms of the potential outcomes, $\mathrm{Y_t}$, of treatment levels $\mathrm{t} \in \{0, 1\}$. Given observable covariates, $\mathbf{X}$, the CATE is defined as the expected difference between the potential outcomes at the measured value $\mathbf{X} = \mathbf{x}$: $\tau(\mathbf{x}) = \mathbb{E}[\mathrm{Y}_1 - \mathrm{Y}_0 \mid \mathbf{X} = \mathbf{x}]$. This causal quantity is fundamentally unidentifiable from observational data without further assumptions because it is not possible to observe both Y_1 and Y_0 for a given unit. However, under the assumptions of consistency, non-interference, ignoreability, and positivity, the CATE is identifiable as the statistical quantity $\tilde{\tau}(\mathbf{x}) = \mathbb{E}[\mathrm{Y} \mid \mathrm{T} = 1, \mathbf{X} = \mathbf{x}] - \mathbb{E}[\mathrm{Y} \mid \mathrm{T} = 0, \mathbf{X} = \mathbf{x}]$ [Rubin, 1980].

Jesson et al. [2021] define BALD acquisition functions for active learning CATE functions from observational data when the cost of acquiring an outcome, y, for a given covariate and treatment pair, $(\mathbf{x}, \mathrm{t})$, is high. Because we do not have labels for Y_1 and Y_0 for each $(\mathbf{x}, \mathrm{t})$ pair in the dataset, their acquisition function focuses on acquiring data points $(\mathbf{x}, \mathrm{t})$ for which it is likely that a matched pair $(\mathbf{x}, 1 - \mathrm{t})$ exists in the pool data or has already been acquired at a previous step. We follow their experiments on their synthetic dataset with limited positivity and the semisynthetic IHDP dataset [Hill, 2011]. Details of the experimental setup are given in [Jesson et al., 2021], we use their provided code, and implement the power acquisition function.

The settings for causal inference experiments are identical to those used in Jesson et al. [2021], using the IHDP dataset [Hill, 2011]. Like them, we use a Deterministic

Uncertainty Estimation Model [van Amersfoort et al., 2021], which is initialized with 100 data points and acquire 10 data points per acquisition batch for 38 steps. The dataset has 471 pool points and a 201 point validation set.

E.2.2 Repeated-MNIST

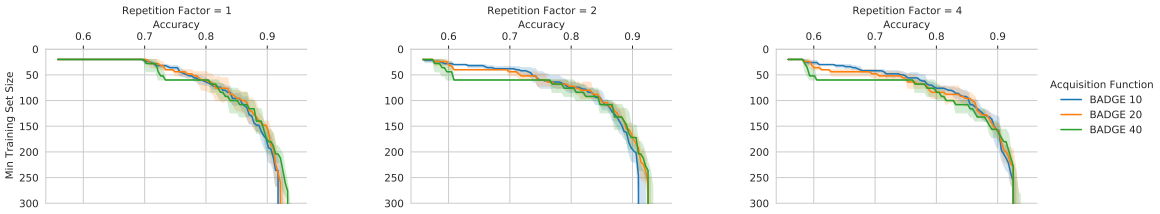


Figure E.2: *Repeated-MNIST x4 (5 trials): acquisition size ablation for BADGE.* Acquisition size 20 performs best out of $\{10, 20, 40\}$. Hence, we use that for Figure 5.2.

BADGE Ablation. In Figure E.2, we see that BADGE performs best with acquisition size 20 on Repeated-MNISTx4 overall. BADGE 40 and BADGE 20 have the highest final accuracy, cf. BADGE 10 while BADGE 20 performs better than BADGE 40 for small training set sizes.

E.2.2.1 Other Scoring Functions

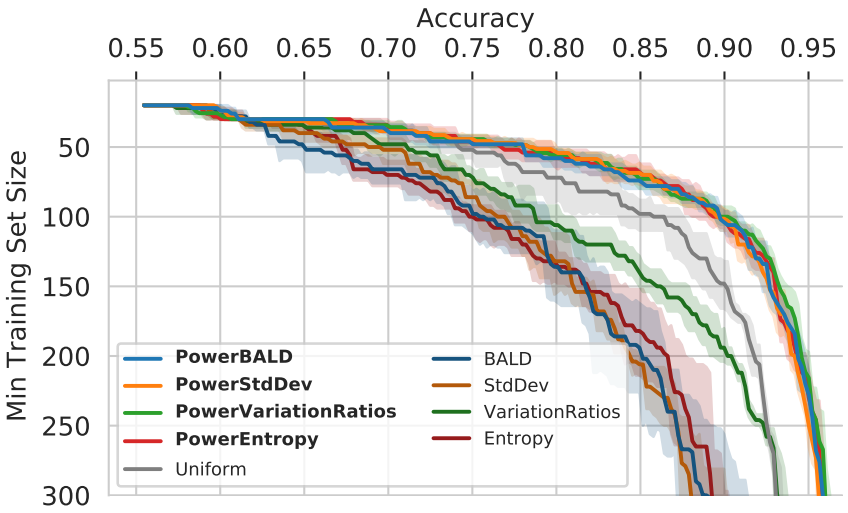


Figure E.3: *Repeated-MNIST x4 (5 trials): Performance for other scoring functions.* Entropy, std dev, variation ratios behave like BALD when applying our stochastic sampling scheme.

In Figure E.3 shows the performance of other scoring functions than BALD on Repeated-MNIST x4.

E.2.2.2 Redundancy Ablation

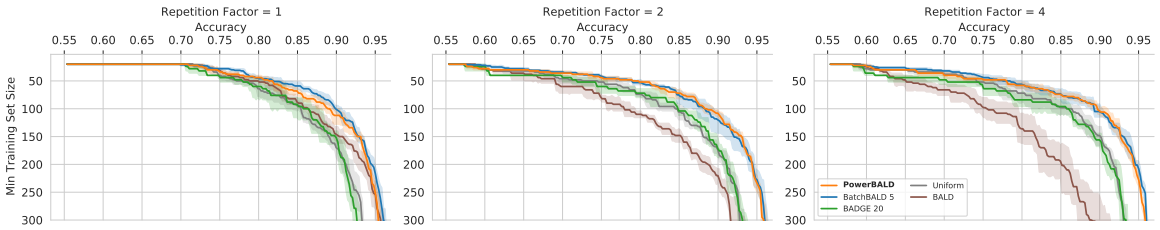
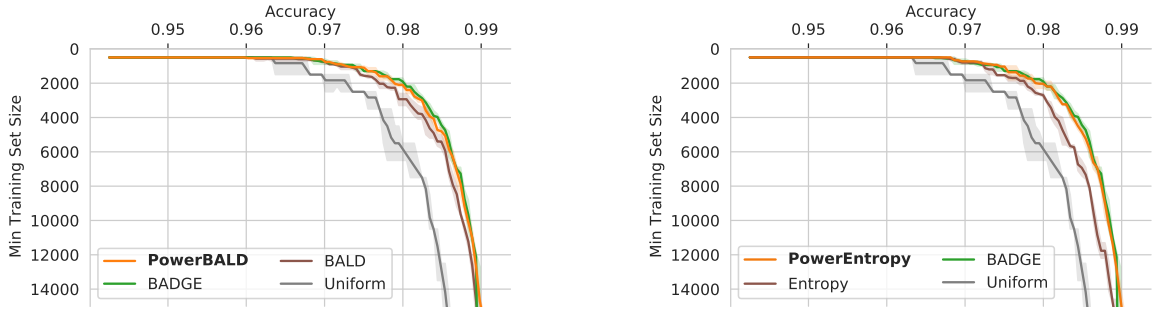


Figure E.4: *Repeated-MNIST (5 trials): Performance ablation for different repetition counts.*

In Figure E.4, we see the same behavior in an ablation for different repetition sizes of Repeated-MNIST.

E.2.3 MIO-TCD



(a) BALD **(b)** Entropy

Figure E.5: *MIO-TCD (5 trials).*

In Figure E.5, we see that power acquisition performs on par with BADGE with both BALD and entropy as underlying score functions.

E.2.4 EMNIST

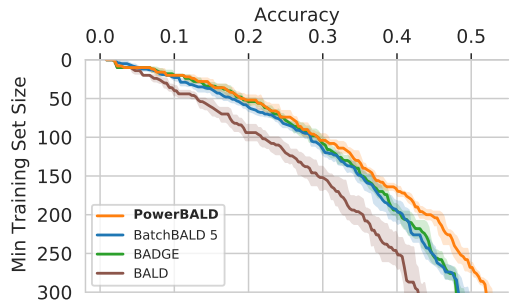


Figure E.6: *EMNIST (Balanced) (5 trials): Performance with BALD.*

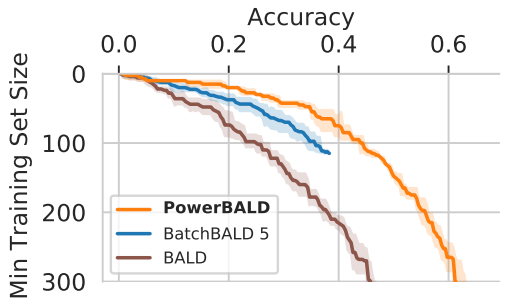


Figure E.7: *EMNIST (ByMerge) (5 trials): Performance with BALD.*

In Figure E.6 and E.7, we see that PowerBALD outperforms BALD, BatchBALD, and BADGE.

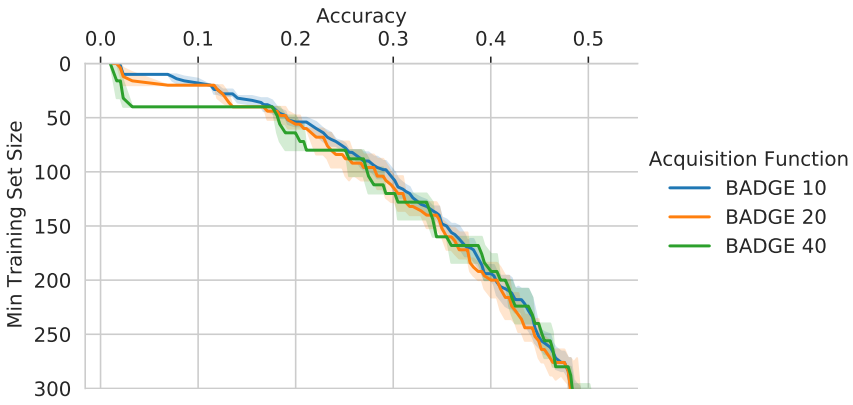


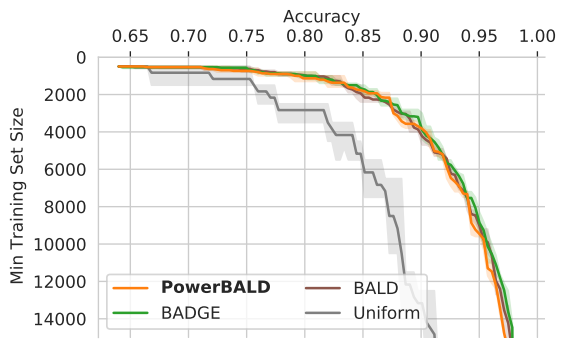
Figure E.8: *EMNIST (Balanced) (5 trials): acquisition size ablation for BADGE.*

BADGE Ablation. In Figure E.8, we see that BADGE performs similarly with all three acquisition sizes. Acquisition size 10 is the smoothest.

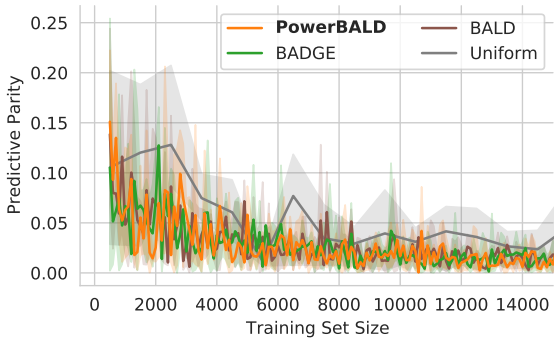
E.2.5 Edge Cases in Synbols

We use Synbols [Lacoste et al., 2020] to demonstrate the behavior of batch active learning in artificially constructed edge cases. Synbols is a character dataset generator for classification where a user can specify the type and proportion of bias and insert artifacts, backgrounds, masking shapes, and so on. We selected three datasets with strong biases supplied by Lacoste et al. [2020]; Branchaud-Charron et al. [2021] to evaluate our method. The experimental settings are described in appendix E.2.1.

For these tasks, performance evaluation includes 'predictive parity', also known as 'accuracy difference', which is the maximum difference in accuracy between subgroups—which are, in this case, different colored characters. This measure is used most widely in domain adaptation and ethics [Verma and Rubin, 2018]. We want to maximize the accuracy while minimizing the predictive parity.



(a) Accuracy

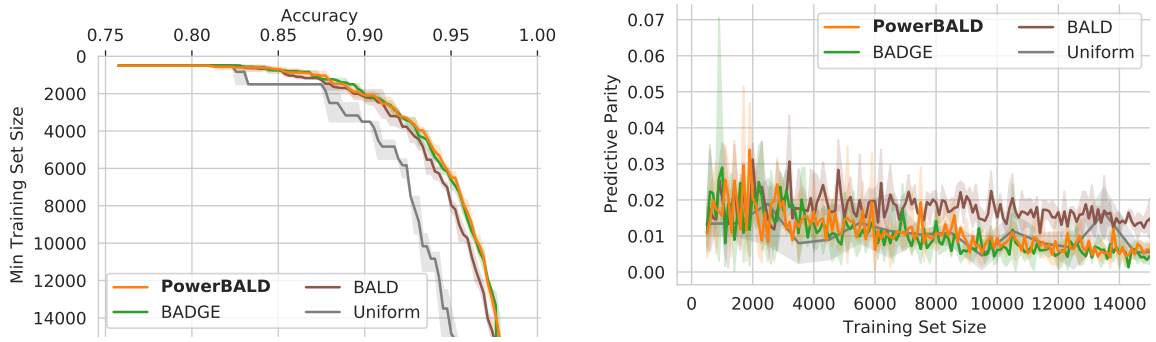


(b) Predictive parity (**Down and left is better.**)

Figure E.9: *Performance on Synbols Spurious Correlations (3 trials) with BALD.* Stochastic acquisition matches BADGE and BALD's predictive parity and performance, which is reassuring as stochastic acquisition functions might be affected by spurious correlations.

Spurious Correlations. This dataset includes spurious correlations between character color and class. As shown in Branchaud-Charron et al. [2021], active learning is especially strong here as characters that do not follow the correlation will be informative and thus selected.

We compare the predictive parity between methods in Fig. E.9(b). We do not see any significant difference between our method and BADGE or BALD. This is encouraging, as stochastic approaches might select more examples following the spurious correlation and thus have higher predictive parity, but this is not the case.



(a) Accuracy **(b)** Predictive parity

Figure E.10: *Synbols Minority Groups (3 trials): Performance on BALD.* PowerBALD outperforms BALD and matches BADGE for both accuracy and predictive parity.

Minority Groups. This dataset includes a subgroup of the data that is under-represented; specifically, most characters are red while few are blue. As Branchaud-Charron et al. [2021] shows, active learning can improve the accuracy for these groups.

Our stochastic approach lets batch acquisition better capture under-represented subgroups. In Figure E.10(a), PowerBALD has an accuracy almost identical to that of BADGE, despite being much cheaper, and outperforms BALD. At the same time, we see in Figure E.10(b) that PowerBALD has a lower predictive parity than BALD, demonstrating a fairer predictive distribution given the unbalanced dataset.

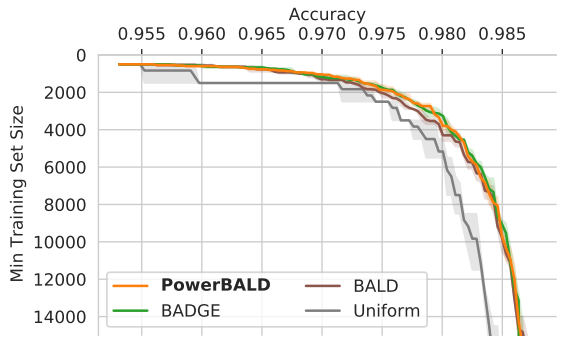


Figure E.11: BALD

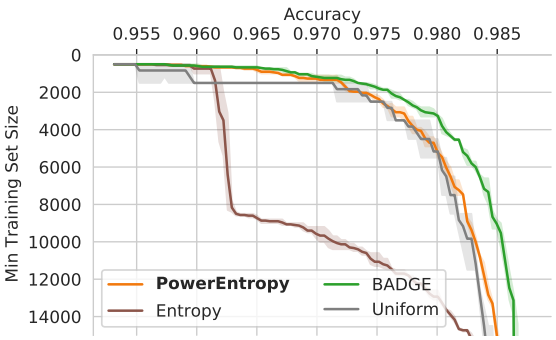


Figure E.12: Entropy

Figure E.13: *Performance on Synbols Missing Characters (3 trials).* In this dataset with high aleatoric uncertainty, PowerBALD matches BADGE and BALD performance. PowerEntropy significantly outperforms Entropy which confounds aleatoric and epistemic uncertainty.

Missing Synbols. This dataset has high aleatoric uncertainty. Some images are missing information required to make high-probability predictions—these images have shapes randomly occluding the character—so even a perfect model would remain uncertain. Lacoste et al. [2020] demonstrated that entropy is ineffective on this data as it cannot distinguish between aleatoric and epistemic uncertainty, while BALD can do so. As a consequence, entropy will unfortunately prefer samples with occluded characters, resulting in degraded active learning performance. For predictive entropy, stochastic acquisition largely corrects the failure of entropy acquisition to account for missing data (Figure E.13) although PowerEntropy still underperforms BADGE here. For BALD, we show in Figure E.11 in the appendix that, as before, our stochastic method performs on par with BADGE and marginally better than BALD.

E.2.6 CLINC-150

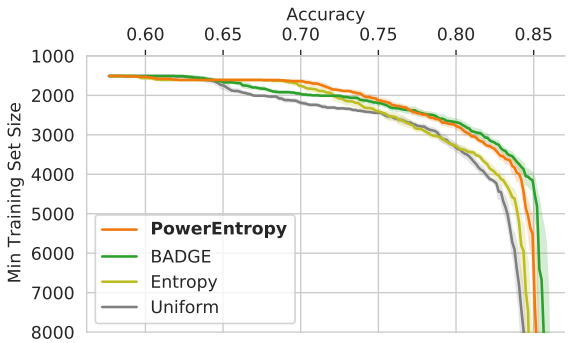


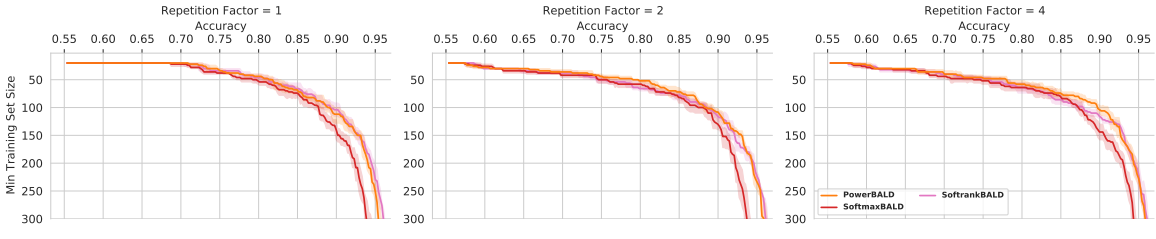
Figure E.14: *Performance on CLINC-150 (10 trials).* PowerEntropy performs much better than entropy, which only performs marginally better than uniform, and almost on par with BADGE.

In Figure E.14, we see that PowerEntropy performs much better than entropy which only performs marginally better than the uniform baseline. PowerEntropy also performs better than BADGE at low training set sizes, but BADGE performs better in the second

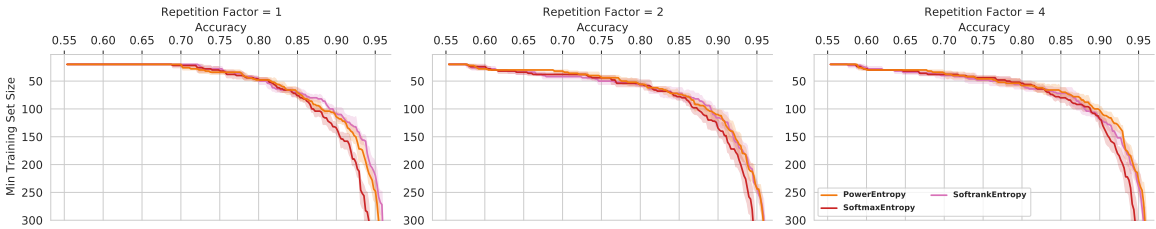
half. Between ≈ 2300 and 4000 samples, BADGE and PowerEntropy perform the same.

E.3 Comparing Power, Softmax and Soft-Rank

E.3.1 Empirical Evidence



(a) BALD



(b) Entropy

Figure E.15: *Repeated-MNIST (5 trials): Performance with all three stochastic strategies.*

Repeated-MNIST. In Figure E.15, power acquisition performs the best overall, followed by soft-rank and then softmax.

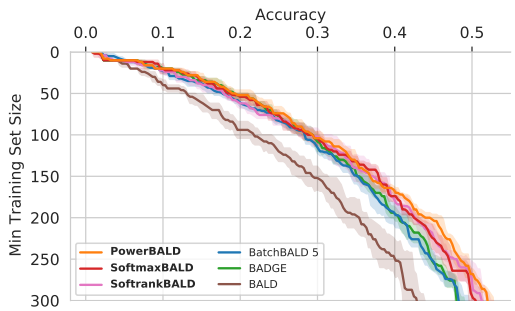


Figure E.16: *EMNIST (Balanced) (5 trials): Performance with all three stochastic strategies with BALD.* PowerBALD performs best.

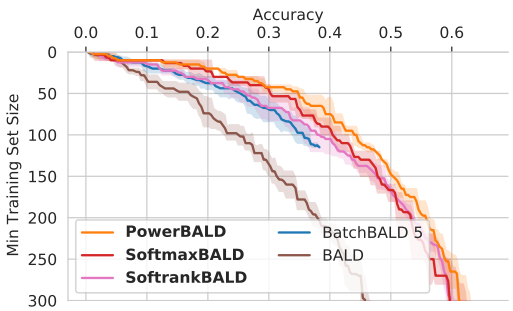
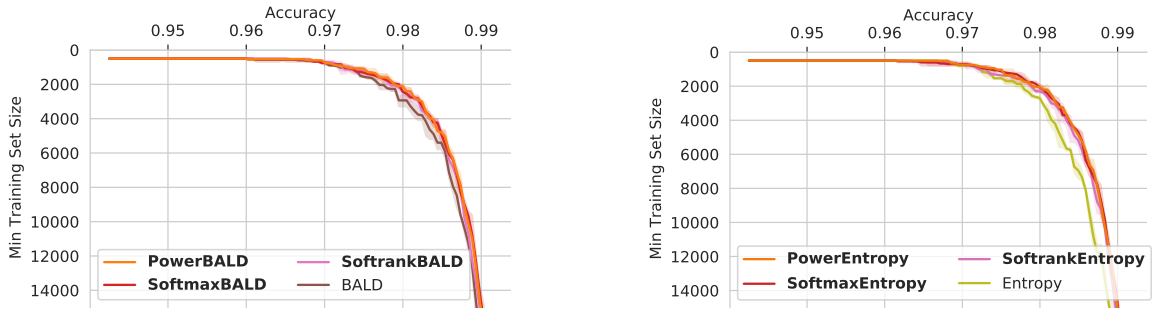


Figure E.17: *EMNIST (ByMerge) (5 trials): Performance with all three stochastic strategies with BALD.* PowerBALD performs best.

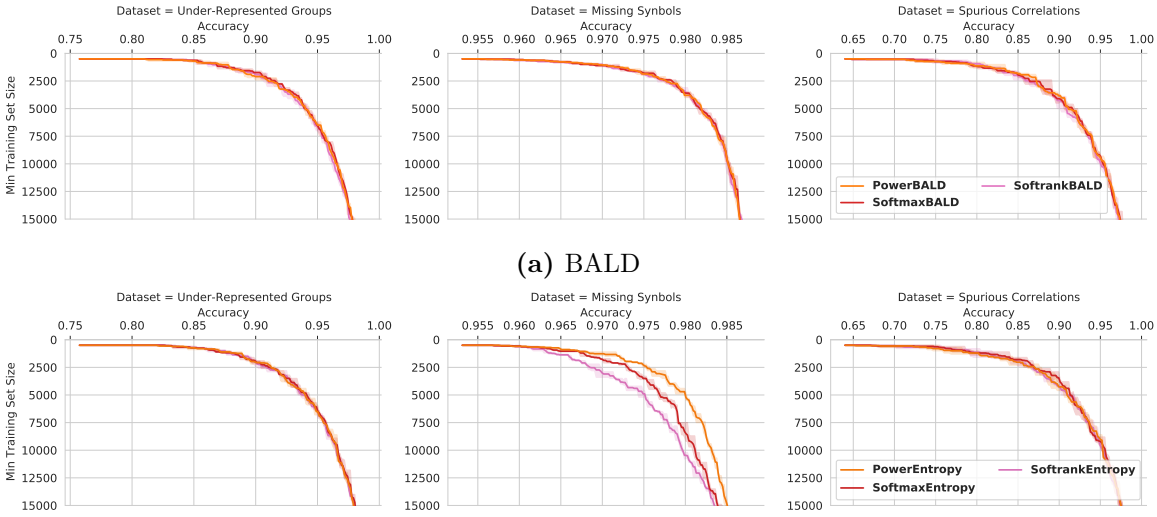
EMNIST. In Figure E.16 and E.17, we see that PowerBALD performs best, but Softmax- and SoftrankBALD also outperform other methods. BADGE did not run on EMNIST (ByMerge) due to out-of-memory issues and BatchBALD took very long as EMNIST (ByMerge) has more than 800,000 samples.



(a) BALD **(b)** Entropy

Figure E.18: *MIO-TCD (3 trials): Performance with all three stochastic strategies.*

MIO-TCD. In Figure E.18, we see that all three stochastic acquisition methods perform about equally well.



(a) BALD

(b) Entropy

Figure E.19: *Synbols edge cases (3 trials): Performance with all three stochastic strategies.*

Synbols. In Figure E.19, power acquisition seems to perform better overall—mainly due to the performance in Synbols Missing Characters.

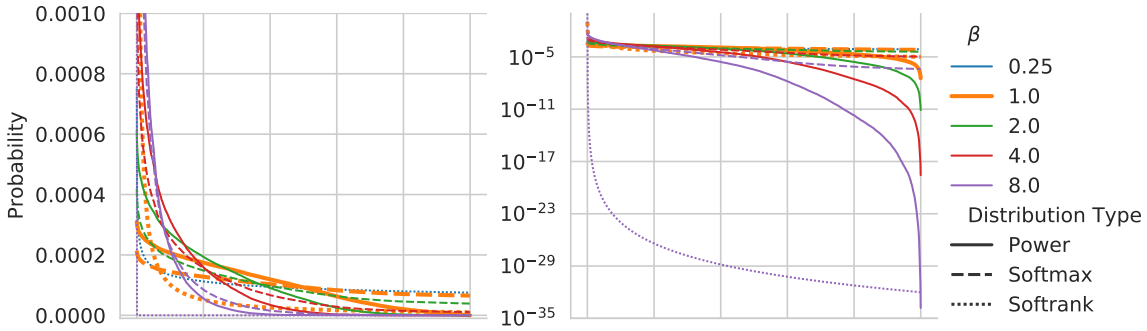


Figure E.21: *Score distribution for power and softmax acquisition of BALD scores on MNIST for varying Coldness β at $t = 0$.* Linear and log plot over samples sorted by their BALD score. At $\beta = 8$ both softmax and power acquisition have essentially the same distribution for high scoring points (closely followed by the power distribution for $\beta = 4$). This might explain why the coldness ablation shows that these β to have very similar AL trajectories on MNIST. Yet, while softmax and power acquisition seem transfer to RMNIST, this is not the case for softrank which is much more sensitive to β. At the same time, power acquisition avoids low-scoring points more than softmax acquisition.

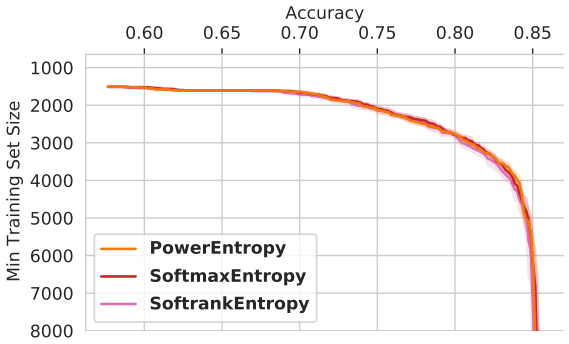


Figure E.20: *CLINC-150 (10 trials): Performance with all three stochastic strategies.*

CLINC-150. In Figure E.20, all three stochastic methods perform similarly.

E.3.2 Investigation

To further examine the three stochastic acquisition variants, we plot their score distributions, extracted from the same MNIST toy example, in Figure E.21. Power and softmax acquisition distributions are similar for $\beta = 8$ (power, softmax) and $\beta = 4$ (softmax). This might explain why active learning with these β shows similar accuracy trajectories.

We find that power and softmax acquisition are quite insensitive to β and thus selecting $\beta = 1$ might generally work quite well.

E.4 Effect of Changing β

E.4.1 Repeated-MNIST

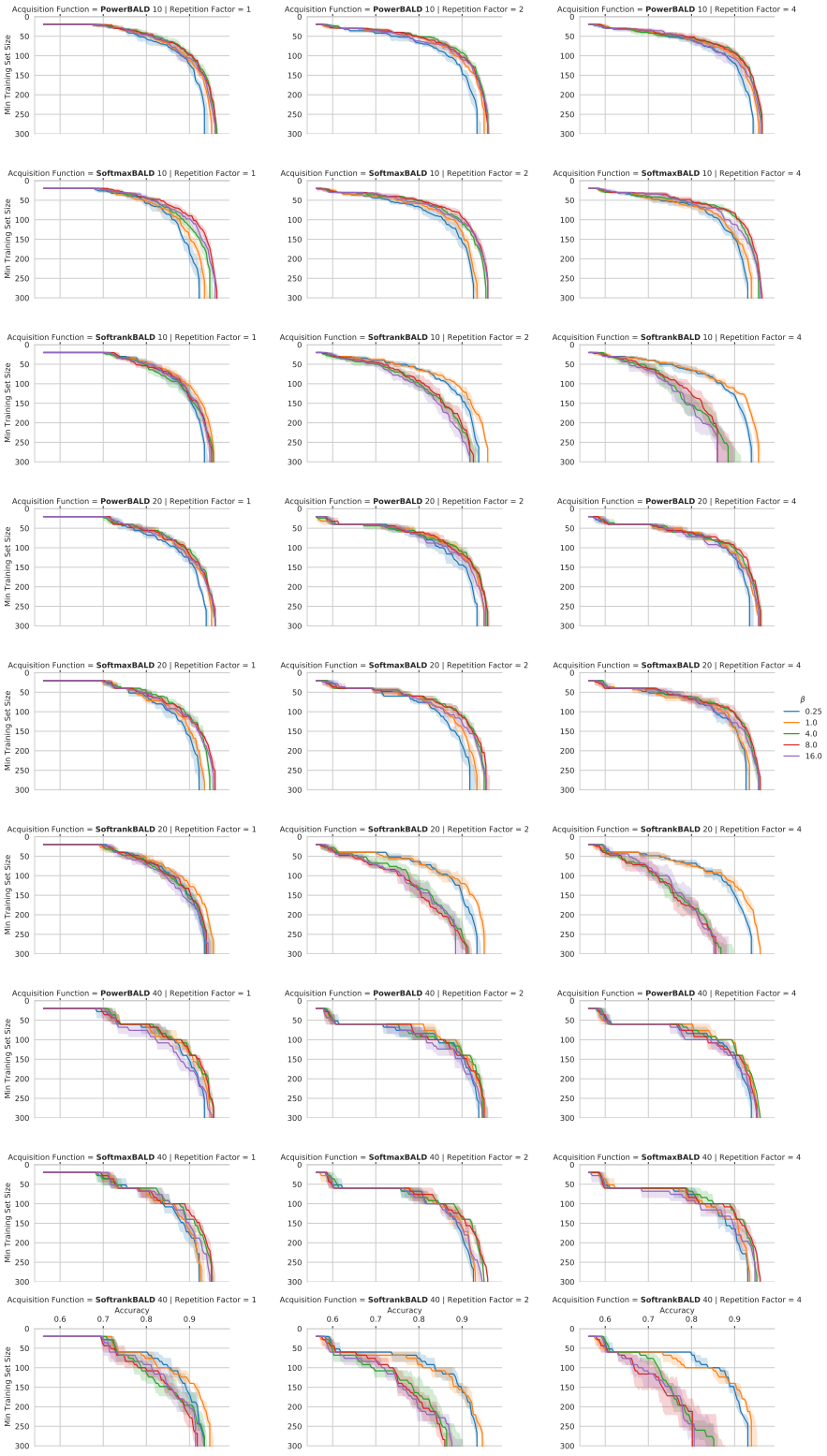


Figure E.22: *Repeated-MNIST: β ablation for *BALD.*

E.4.1.1 MIO-TCD and Synbols

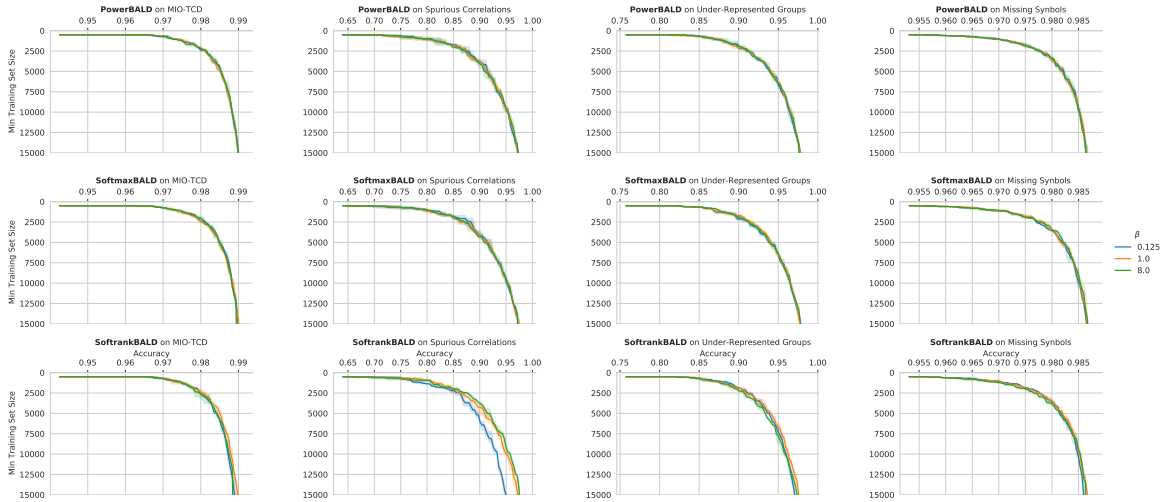


Figure E.23: *MIO-TCD and Synbols: β ablation for *BALD.*

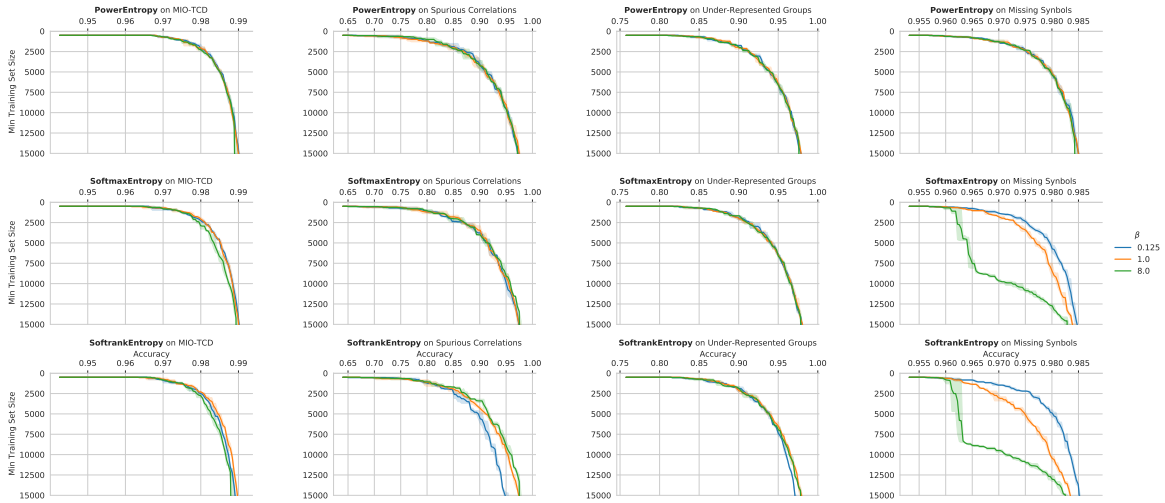
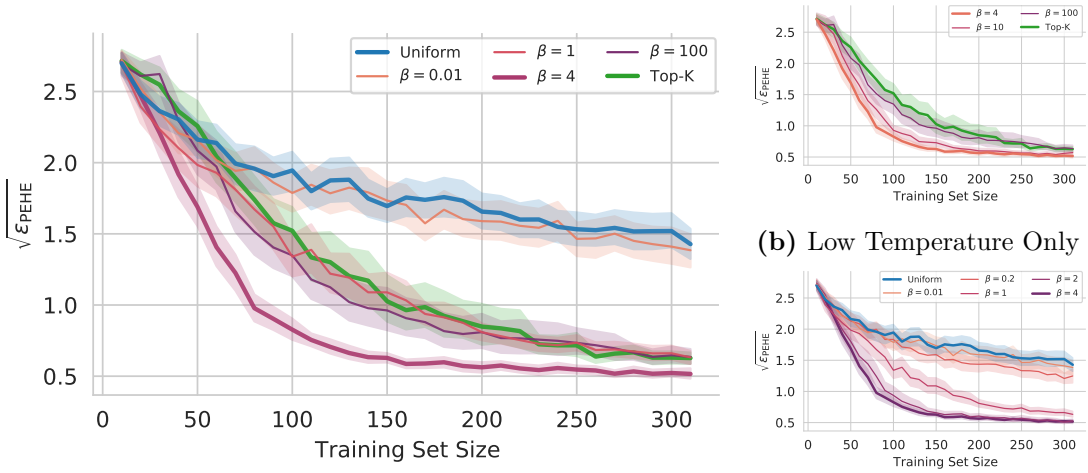


Figure E.24: *MIO-TCD and Synbols: β ablation for *Entropy.*

E.4.2 CausalBALD: Synthetic Dataset



(a) Overall Ablation (Subset)

(b) Low Temperature Only

(c) High Temperature Only

Figure E.25: *CausalBALD: Synthetic Dataset.* (a) At a very high temperature ($\beta = 0.1$), PowerBALD behaves very much like random acquisition, and as the temperature decreases the performance of the acquisition function improves (lower $\sqrt{\epsilon_{\text{PEHE}}}$). (b) Eventually, the performance reaches an inflection point ($\beta = 4.0$) and any further decrease in temperature results in the acquisition strategy performing more like top-K. We see that under the optimal temperature, power acquisition significantly outperforms both random acquisition and top-K over a wide range of temperature settings.

We provide further β ablations for CausalBALD on the entirely synthetic dataset which is used by Jesson et al. [2021]. This demonstrates the ways in which β interpolates between uniform and top-K acquisition.

E.4.3 CLINC-150

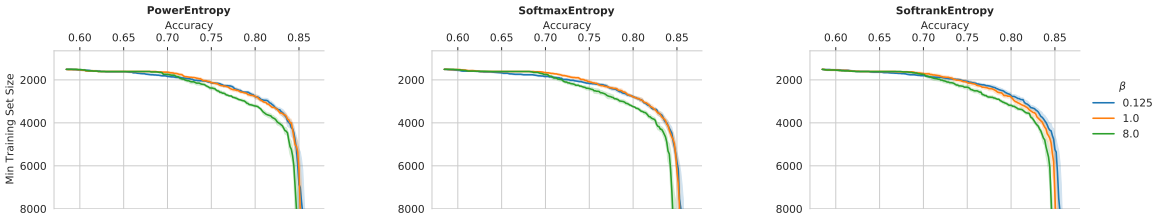


Figure E.26: Performance CLINC-150: β ablation for *Entropy.

F

Prediction- & Distribution-Aware Bayesian Active Learning

F.1 BALD Estimation

In general, we can estimate BALD using nested Monte Carlo [Rainforth et al., 2018]:

$$\mathrm{BALD}(\boldsymbol{x}) = \mathbb{E}_{\mathrm{p}(\theta)}[-\mathbb{E}_{\mathrm{p}(y|\boldsymbol{x})}[\log \mathrm{p}(y \mid \boldsymbol{x})] + \mathbb{E}_{\mathrm{p}(y|\boldsymbol{x},\theta)}[\log \mathrm{p}(y \mid \boldsymbol{x}, \theta)]] \tag{F.1}$$

$$\approx \frac{1}{M}\sum_{j=1}^{M} -\log\left(\frac{1}{K}\sum_{i=1}^{K}\mathrm{p}(y_j \mid \boldsymbol{x}, \theta_i)\right) + \log \mathrm{p}(y_j \mid \boldsymbol{x}, \theta_j), \tag{F.2}$$

where $\theta_i \sim \mathrm{p}(\theta)$, $(\theta_j, y_j) \sim \mathrm{p}(\theta)\,\mathrm{p}(y \,|\, \boldsymbol{x}, \theta)$. Special cases allow us to use computationally cheaper estimators.

F.1.1 Categorical Predictive Distribution

When y and y^{eval} are discrete, we can write

$$\mathrm{BALD}(\boldsymbol{x}) = \mathbb{E}_{\mathrm{p}(\theta)}[-\mathbb{E}_{\mathrm{p}(y|\boldsymbol{x})}[\log \mathrm{p}(y \mid \boldsymbol{x})] + \mathbb{E}_{\mathrm{p}(y|\boldsymbol{x},\theta)}[\log \mathrm{p}(y \mid \boldsymbol{x}, \theta)]] \tag{F.3}$$

$$= -\mathbb{E}_{\mathrm{p}(y|\boldsymbol{x})}[\log \mathrm{p}(y \mid \boldsymbol{x})] + \mathbb{E}_{\mathrm{p}(\theta)\,\mathrm{p}(y|\boldsymbol{x},\theta)}[\log \mathrm{p}(y \mid \boldsymbol{x}, \theta)] \tag{F.4}$$

$$= -\sum_{y\in\mathcal{Y}} \mathrm{p}(y \mid \boldsymbol{x}) \log \mathrm{p}(y \mid \boldsymbol{x}) + \mathbb{E}_{\mathrm{p}(\theta)}[\sum_{y\in\mathcal{Y}} \mathrm{p}(y \mid \boldsymbol{x}, \theta) \log \mathrm{p}(y \mid \boldsymbol{x}, \theta)]. \tag{F.5}$$

This can be estimated using samples, $\theta_i \sim \mathrm{p}(\theta)$ [Houlsby, 2014]:

$$\mathrm{BALD}(\boldsymbol{x}) \approx -\sum_{y\in\mathcal{Y}} \hat{\mathrm{p}}(y \mid \boldsymbol{x}) \log \hat{\mathrm{p}}(y \mid \boldsymbol{x}) + \frac{1}{K}\sum_{i=1}^{K}\sum_{y\in\mathcal{Y}} \mathrm{p}(y \mid \boldsymbol{x}, \theta_i) \log \mathrm{p}(y \mid \boldsymbol{x}, \theta_i), \tag{F.6}$$

where

$$\hat{\mathrm{p}}(y \mid \boldsymbol{x}) = \frac{1}{K}\sum_{i=1}^{K} \mathrm{p}(y \mid \boldsymbol{x}, \theta_i). \tag{F.7}$$

F.1.2 Gaussian Predictive Distribution

Suppose we have a model whose likelihood function, $\mathrm{p}(y \mid \boldsymbol{x}, \theta)$, and predictive distribution, $\mathrm{p}(y|\boldsymbol{x})$, are Gaussian. Then, using the symmetry of the mutual information along with knowledge of the entropy of a Gaussian [Cover and Thomas, 2005], we have

$$\mathrm{BALD}(\boldsymbol{x}) = \frac{1}{2}\log 2\pi e\,\mathrm{Var}[Y \mid \boldsymbol{x}] - \frac{1}{2}\,\mathbb{E}_{\mathrm{p}(\theta)}[\log 2\pi e\,\mathrm{Var}[Y \mid \boldsymbol{x}, \theta]] \tag{F.8}$$

$$= \frac{1}{2}\left(\log \operatorname{Var}[Y \mid \boldsymbol{x}] - \mathbb{E}_{\mathrm{p}(\theta)}[\log \operatorname{Var}[Y \mid \boldsymbol{x}, \theta]]\right). \tag{F.9}$$

Relatedly, Houlsby et al. [2011] identified a closed-form approximation of BALD for the particular case of using a probit likelihood function, a Gaussian-process prior and a Gaussian approximation to the predictive distribution.

F.2 EPIG Derivation

Computing an expectation over both y and $\boldsymbol{x}^{\text{eval}}$ gives the expected predictive information gain:

$$\begin{aligned}
\operatorname{EPIG}(\boldsymbol{x}) &= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})\,\mathrm{p}(y|\boldsymbol{x})}[\mathrm{H}(\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}})) - \mathrm{H}(\mathrm{p}(y^{\text{eval}} \mid y, \boldsymbol{x}, \boldsymbol{x}^{\text{eval}}))] && \text{(F.10)}\\
&= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})\,\mathrm{p}(y|\boldsymbol{x})}[-\mathbb{E}_{\mathrm{p}(y^{\text{eval}}|\boldsymbol{x}^{\text{eval}})}[\log \mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}})]] \\
&\quad + \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})\,\mathrm{p}(y|\boldsymbol{x})}[\mathbb{E}_{\mathrm{p}(y^{\text{eval}}|y,\boldsymbol{x},\boldsymbol{x}^{\text{eval}})}[\log \mathrm{p}(y^{\text{eval}} \mid y, \boldsymbol{x}, \boldsymbol{x}^{\text{eval}})]] && \text{(F.11)}\\
&= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})\,\mathrm{p}(y,y^{\text{eval}}|\boldsymbol{x},\boldsymbol{x}^{\text{eval}})}[\log \frac{\mathrm{p}(y^{\text{eval}} \mid y, \boldsymbol{x}, \boldsymbol{x}^{\text{eval}})}{\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}})}] && \text{(F.12)}\\
&= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})\,\mathrm{p}(y,y^{\text{eval}}|\boldsymbol{x},\boldsymbol{x}^{\text{eval}})}[\log \frac{\mathrm{p}(y \mid \boldsymbol{x})\,\mathrm{p}(y^{\text{eval}} \mid y, \boldsymbol{x}, \boldsymbol{x}^{\text{eval}})}{\mathrm{p}(y \mid \boldsymbol{x})\,\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}})}] && \text{(F.13)}\\
&= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})\,\mathrm{p}(y,y^{\text{eval}}|\boldsymbol{x},\boldsymbol{x}^{\text{eval}})}[\log \frac{\mathrm{p}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\text{eval}})}{\mathrm{p}(y \mid \boldsymbol{x})\,\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}})}] && \text{(F.14)}\\
&= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})}[\mathrm{I}[y; y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\text{eval}}]] && \text{(F.15)}\\
&= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})}[\mathrm{D}_{\mathrm{KL}}(\mathrm{p}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\text{eval}}) \parallel \mathrm{p}(y \mid \boldsymbol{x})\,\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}))]. && \text{(F.16)}
\end{aligned}$$

F.3 EPIG Estimation

While in general we can use Equation 7.39 to estimate EPIG, special cases allow computationally cheaper estimators.

F.3.1 Categorical Predictive Distribution

When y and y^{eval} are discrete, we can write

$$\begin{aligned}
\operatorname{EPIG}(\boldsymbol{x}) &= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})}[\mathrm{D}_{\mathrm{KL}}(\mathrm{p}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\text{eval}}) \parallel \mathrm{p}(y \mid \boldsymbol{x})\,\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}))] && \text{(F.17)}\\
&= \mathbb{E}_{\mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})}[\sum_{y \in \mathcal{Y}} \sum_{y^{\text{eval}} \in \mathcal{Y}} \mathrm{p}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\text{eval}}) \log \frac{\mathrm{p}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\text{eval}})}{\mathrm{p}(y \mid \boldsymbol{x})\,\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}})}]. && \text{(F.18)}
\end{aligned}$$

This can be estimated using samples, $\theta_i \sim \mathrm{p}(\theta)$ and $\boldsymbol{x}_j^{\text{eval}} \sim \mathrm{p}_{\text{eval}}(\boldsymbol{x}^{\text{eval}})$:

$$\operatorname{EPIG}(\boldsymbol{x}) \approx \frac{1}{M} \sum_{j=1}^{M} \sum_{y \in \mathcal{Y}} \sum_{y^{\text{eval}} \in \mathcal{Y}} \hat{\mathrm{p}}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}_j^{\text{eval}}) \log \frac{\hat{\mathrm{p}}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}_j^{\text{eval}})}{\hat{\mathrm{p}}(y \mid \boldsymbol{x}) \hat{\mathrm{p}}(y^{\text{eval}} \mid \boldsymbol{x}_j^{\text{eval}})}, \tag{F.19}$$

where

$$\hat{\mathrm{p}}(y, y^{\text{eval}} \mid \boldsymbol{x}, \boldsymbol{x}_j^{\text{eval}}) = \frac{1}{K} \sum_{i=1}^{K} \mathrm{p}(y \mid \boldsymbol{x}, \theta_i)\,\mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}_j^{\text{eval}}, \theta_i) \tag{F.20}$$

$$\hat{\mathrm{p}}(y \mid \boldsymbol{x}) = \frac{1}{K} \sum_{i=1}^{K} \mathrm{p}(y \mid \boldsymbol{x}, \theta_i) \tag{F.21}$$

$$\hat{\mathrm{p}}(y^{\text{eval}} \mid \boldsymbol{x}_j^{\text{eval}}) = \frac{1}{K} \sum_{i=1}^{K} \mathrm{p}(y^{\text{eval}} \mid \boldsymbol{x}_j^{\text{eval}}, \theta_i). \tag{F.22}$$

F.3.2 Gaussian Predictive Distribution

Consider a joint predictive distribution that is multivariate Gaussian with mean vector μ and covariance matrix Σ:

$$\mathrm{p}(y, y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\mathrm{eval}}) = \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\mu, \begin{bmatrix} \mathrm{Cov}[x; x] & \mathrm{Cov}[x; \boldsymbol{x}^{\mathrm{eval}}] \\ \mathrm{Cov}[x; \boldsymbol{x}^{\mathrm{eval}}] & \mathrm{Cov}[\boldsymbol{x}^{\mathrm{eval}}; \boldsymbol{x}^{\mathrm{eval}}] \end{bmatrix}\right). \tag{F.23}$$

In this setting the mutual information between y and y^{eval} given $\boldsymbol{x}$ and $\boldsymbol{x}^{\mathrm{eval}}$ is a closed-form function of Σ:

$$\begin{aligned}
\mathrm{I}[Y; Y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{X}^{\mathrm{eval}}] &= \mathrm{H}(\mathrm{p}(y \mid \boldsymbol{x})) + \mathrm{H}(\mathrm{p}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}})) - \mathrm{H}(\mathrm{p}(y, y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\mathrm{eval}})) && \text{(F.24)} \\
&= \frac{1}{2} \log 2\pi e \, \mathrm{Var}[\mathrm{p}(y \mid \boldsymbol{x})] + \frac{1}{2} \log 2\pi e \, \mathrm{Var}[\mathrm{p}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}})] \\
&= -\frac{1}{2} \log \det 2\pi e \Sigma && \text{(F.25)} \\
&= \frac{1}{2} \log \frac{\mathrm{Var}[\mathrm{p}(y \mid \boldsymbol{x})] \, \mathrm{Var}[\mathrm{p}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}})]}{\det \Sigma} && \text{(F.26)} \\
&= \frac{1}{2} \log \frac{\mathrm{Cov}[x; x] \, \mathrm{Cov}[\boldsymbol{x}^{\mathrm{eval}}; \boldsymbol{x}^{\mathrm{eval}}]}{\det \Sigma} && \text{(F.27)} \\
&= \frac{1}{2} \log \frac{\mathrm{Cov}[x; x] \, \mathrm{Cov}[\boldsymbol{x}^{\mathrm{eval}}; \boldsymbol{x}^{\mathrm{eval}}]}{\mathrm{Cov}[x; x] \, \mathrm{Cov}[\boldsymbol{x}^{\mathrm{eval}}; \boldsymbol{x}^{\mathrm{eval}}] - \mathrm{Cov}[x; \boldsymbol{x}^{\mathrm{eval}}]^2}. && \text{(F.28)}
\end{aligned}$$

We can estimate EPIG using samples, $\boldsymbol{x}_j^{\mathrm{eval}} \sim \mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})$:

$$\begin{aligned}
\mathrm{EPIG}(\boldsymbol{x}) &= \mathbb{E}_{\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}})}[\mathrm{I}[y; y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\mathrm{eval}}]] \approx \frac{1}{M} \sum_{j=1}^{M} \mathrm{I}[y; y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{x}_j^{\mathrm{eval}}] && \text{(F.29)} \\
&= \frac{1}{2M} \sum_{j=1}^{M} \log \frac{\mathrm{Cov}[x; x] \, \mathrm{Cov}[\boldsymbol{x}_j^{\mathrm{eval}}; \boldsymbol{x}_j^{\mathrm{eval}}]}{\mathrm{Cov}[x; x] \, \mathrm{Cov}[\boldsymbol{x}_j^{\mathrm{eval}}; \boldsymbol{x}_j^{\mathrm{eval}}] - \mathrm{Cov}[x; \boldsymbol{x}_j^{\mathrm{eval}}]^2}. && \text{(F.30)}
\end{aligned}$$

F.3.3 Connection to Foster et al. [2019]

Foster et al. [2019] primarily considered variational estimation of the expected information gain. Since the joint density, $\mathrm{p}(y, y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\mathrm{eval}})$, that appears in EPIG is often not known in closed form, EPIG estimation broadly falls under the "implicit likelihood" category of methods considered in that paper. Here, we focus on showing how the "posterior" or Barber-Agakov bound [Barber and Agakov, 2003] from this earlier work applies to EPIG estimation. We first recall Equation 7.21,

$$\mathrm{EPIG}(\boldsymbol{x}) = \mathbb{E}_{\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}}) \, \mathrm{p}(y, y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\mathrm{eval}})}[\log \mathrm{p}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, y, \boldsymbol{x})] + \mathrm{H}(\mathrm{p}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}})), \tag{F.31}$$

and the observation that $c = \mathrm{H}(\mathrm{p}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}))$ does not depend upon $\boldsymbol{x}$ and hence can be neglected when choosing between designs. By Gibbs's inequality, we must have

$$\mathrm{EPIG}(\boldsymbol{x}) \geq \mathbb{E}_{\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}}) \, \mathrm{p}(y, y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\mathrm{eval}})}[\log q(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, y, \boldsymbol{x})] + \mathrm{H}(\mathrm{p}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}})) \tag{F.32}$$

for any distribution q. We can now consider a variational family, $q_\psi(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, y, \boldsymbol{x})$, and a maximization over the variational parameter, ψ:

$$\mathrm{EPIG}(\boldsymbol{x}) \geq \sup_{\psi} \mathbb{E}_{\mathrm{p}_{\mathrm{eval}}(\boldsymbol{x}^{\mathrm{eval}}) \, \mathrm{p}(y, y^{\mathrm{eval}} \mid \boldsymbol{x}, \boldsymbol{x}^{\mathrm{eval}})}[\log q_\psi(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, y, \boldsymbol{x})] + \mathrm{H}(\mathrm{p}(y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}})). \tag{F.33}$$

A practical implication of this bound is that we could estimate EPIG by learning an auxiliary network, $q_\psi(y^{\text{eval}}|\boldsymbol{x}^{\text{eval}}, y, \boldsymbol{x})$, using data simulated from the model to make one-step-ahead predictions. That is, q_ψ is trained to make predictions at $\boldsymbol{x}^{\text{eval}}$, incorporating the knowledge of the hypothetical acquisition $(\boldsymbol{x}, y)$. For our purposes, training such an auxiliary network at each acquisition is prohibitively expensive. But this approach might be valuable in other applications of EPIG.

F.4 Dataset Construction

F.4.1 UCI Data

For each dataset we start by taking the base dataset, $\mathcal{D}_{\text{base}}$, from the UCI repository. Satellite and Vowels have predefined test datasets, $\mathcal{D}_{\text{test}}$. In contrast, Magic does not have a predefined train-test split. It is stated in Magic's documentation that one of the classes is underrepresented in the dataset relative to real-world data (Magic is a simulated dataset). Whereas classes 0 and 1 respectively constitute 65% and 35% of the dataset, it is stated that class 1 constitutes the majority of cases in reality (the exact split is not stated; we assume 75% for class 1). We therefore uniformly sample 30% of $\mathcal{D}_{\text{base}}$ to form a test base dataset, $\mathcal{D}'_{\text{base}}$; then we set $\mathcal{D}_{\text{base}} \leftarrow \mathcal{D}_{\text{base}} \setminus \mathcal{D}'_{\text{base}}$; then we make $\mathcal{D}_{\text{test}}$ by removing input-label pairs from $\mathcal{D}'_{\text{base}}$ such that class 1 constitutes 75% of the subset. With the test set defined, we proceed to sample two disjoint subsets of $\mathcal{D}_{\text{base}}$ such that their class proportions match those of $\mathcal{D}_{\text{base}}$: a pool set, $\mathcal{D}_{\text{pool}}$, whose size varies between datasets, and a validation set, $\mathcal{D}_{\text{val}}$, of 60 input-label pairs. Regardless of the class proportions of $\mathcal{D}_{\text{base}}$, we always use an initial training dataset, $\mathcal{D}_{\text{init}}$, of 2 input-label pairs per class, sampled from $\mathcal{D}_{\text{base}}$. Finally, we sample a representative set of inputs, $\mathcal{D}_*$, whose class proportions match those of $\mathcal{D}_{\text{test}}$.

F.4.2 MNIST Data

Implementing each setting starts by using the standard MNIST training data (60,000 input-label pairs) as the base dataset, $\mathcal{D}_{\text{base}}$, and the standard MNIST testing data (10,000 input-label pairs) as the test base dataset, $\mathcal{D}'_{\text{base}}$. For Redundant MNIST we make $\mathcal{D}_{\text{test}}$ by removing input-label pairs from $\mathcal{D}'_{\text{base}}$ such that only classes 1 and 7 remain. Otherwise, we set $\mathcal{D}_{\text{test}} = \mathcal{D}'_{\text{base}}$. Next we construct the pool set, $\mathcal{D}_{\text{pool}}$. For Curated MNIST and Redundant MNIST we sample 4,000 inputs per class from $\mathcal{D}'_{\text{base}}$. For Unbalanced MNIST we sample 400 inputs per class for classes 0-4 and 4,000 inputs per class for classes 5-9. After this we make the initial training dataset, $\mathcal{D}_{\text{init}}$. For Curated MNIST and Unbalanced MNIST we sample 2 input-label pairs per class from $\mathcal{D}_{\text{base}}$. For Redundant MNIST we sample 2 input-label pairs from class 1, 2 input-label pairs from class 7 and 1 input-label pair per class from 2 randomly selected classes other than 1 and 7. Next, the validation set, $\mathcal{D}_{\text{val}}$. For all settings this comprises 60 input-label pairs such that the class proportions match those used to form $\mathcal{D}_{\text{pool}}$. Finally, we sample a representative set of inputs, $\mathcal{D}_*$, whose class proportions match those of $\mathcal{D}_{\text{test}}$.

F.5 EPIG & JEPIG

Proposition 7.2. *EPIG lower-bounds 'averaged' JEPIG:*

$$\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] \leq {}^1\!/\!_{\mathrm{E}}\, \mathrm{I}[Y^{\text{eval}}_{1..\mathrm{E}}; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{eval}}_{1..\mathrm{E}}, \boldsymbol{x}^{\text{acq}}] + c_{\text{eval}}, \tag{7.54}$$

up to an additive constant (c_{eval}) that only depends on the evaluation samples and is independent of $\boldsymbol{x}^{\text{acq}}$. The inequality gap is the total correlation:

$$1/\mathtt{E}\,\mathrm{TC}[Y_1^{\text{eval}};\ldots;Y_{\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}] \tag{7.55}$$

We have equality when it is zero, that is when the predictions on the evaluation set are independent (given the acquisition samples).

Proof. We drop conditioning on $\mathcal{D}^{\text{train}}$ in this proof. First, we remind ourselves of the total correlation:

$$\mathrm{TC}[A_1;\ldots;A_n] = \sum_{i=1}^{n} \mathrm{H}[A_i] - \mathrm{H}[A_{1,\ldots,n}] \geq 0. \tag{F.34}$$

We expand EPIG to the right:

$$\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] = \mathrm{H}[Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}] - \mathrm{H}[Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}]. \tag{F.35}$$

The first term on the right-hand side is constant given a fixed evaluation set. We can express both terms in terms of the total correlation:

$$\mathtt{E}\;\mathrm{H}[Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}] \tag{F.36}$$

$$= \mathrm{TC}[Y_1^{\text{eval}};\ldots;Y_{\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}] + \mathrm{H}[Y_{1..\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}], \tag{F.37}$$

and

$$\mathtt{E}\;\mathrm{H}[Y^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}] \tag{F.38}$$

$$= \mathrm{TC}[Y_1^{\text{eval}};\ldots;Y_{\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}] + \mathrm{H}[Y_{1..\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}]. \tag{F.39}$$

$c_{\text{eval}} \triangleq 1/\mathtt{E}\,\mathrm{TC}[Y_1^{\text{eval}};\ldots;Y_{\mathtt{E}}^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}]$ is independent of the acquisition set. Hence, we can conclude:

$$\mathrm{I}[Y^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{X}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] \tag{F.40}$$

$$= 1/\mathtt{E}(\mathrm{H}[Y_{1..\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}] - \mathrm{H}[Y_{1..\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}] \tag{F.41}$$

$$- \mathrm{TC}[Y_1^{\text{eval}};\ldots;Y_{\mathtt{E}}^{\text{eval}} \mid \boldsymbol{X}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}]) + c_{\text{eval}} \tag{F.42}$$

$$= 1/\mathtt{E}\,\mathrm{I}[Y_{1..\mathtt{E}}^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] + c_{\text{eval}} \tag{F.43}$$

$$- 1/\mathtt{E}\,\underbrace{\mathrm{TC}[Y_1^{\text{eval}};\ldots;Y_{\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}]}_{\geq 0} \tag{F.44}$$

$$\leq 1/\mathtt{E}\,\mathrm{I}[Y_{1..\mathtt{E}}^{\text{eval}}; Y^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}^{\text{acq}}] + c_{\text{eval}}. \tag{F.45}$$

Thus, we see that the inequality gap is $1/\mathtt{E}\,\mathrm{TC}[Y_1^{\text{eval}};\ldots;Y_{\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, Y^{\text{acq}}, \boldsymbol{x}^{\text{acq}}]$, which is zero, if and only if the random variables $Y_i^{\text{eval}} \mid \boldsymbol{x}_i^{\text{eval}}$ are independent given the acquisition samples. □

F.6 A Practical Approximation of JEPIG

We want to find an approximation $\hat{\boldsymbol{\Omega}}$ with distribution $\mathrm{q}(\hat{\boldsymbol{\omega}})$, such that for all possible acquisition sets, we have:

$$\mathrm{I}[Y_{1..\mathtt{K}}^{\text{acq}}; \boldsymbol{\Omega} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, Y_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}}] \approx \mathrm{I}[Y_{1..\mathtt{K}}^{\text{acq}}; \hat{\boldsymbol{\Omega}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}]. \tag{F.46}$$

We note two properties of this conditional mutual information and the underlying models $\mathrm{p}(\boldsymbol{\Omega} \mid y_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}})$ for different $y_{1..\mathtt{E}}^{\text{eval}} \sim \mathrm{p}(y_{1..\mathtt{E}}^{\text{eval}} \mid \mathcal{D}^{\text{train}})$:

1. marginalizing $\mathrm{p}(\boldsymbol{\Omega} \mid y_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}})$ over all possible $y_{1..\mathtt{E}}^{\text{eval}}$ yields the predictions of the original posterior $\mathrm{p}(\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}})$, so we would like to have
$$\mathbb{E}_{\mathrm{q}(\hat{\boldsymbol{\omega}})}\, \mathrm{p}(y \mid x, \hat{\boldsymbol{\omega}}) = \mathrm{p}(y \mid x, \mathcal{D}^{\text{train}});$$
2. $\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, Y_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}}] \leq \mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$, and when $\boldsymbol{x} \in \{\boldsymbol{x}_i^{\text{eval}}\}_i$, we expect $\mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, Y_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}}] \ll \mathrm{I}[Y; \boldsymbol{\Omega} \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}]$. In other words, the epistemic uncertainty of evaluation samples $\boldsymbol{x}^{\text{eval}}$ ought to decrease when we also train on the evaluation set (using pseudo-labels $y_{1..\mathtt{E}}^{\text{eval}}$), and we would like the same for $\hat{\boldsymbol{\Omega}}$:
$$\mathrm{I}[Y; \hat{\boldsymbol{\Omega}} \mid x] \leq \mathrm{I}[Y; \boldsymbol{\Omega} \mid x, \mathcal{D}^{\text{train}}].$$

Note, that the second property follows from EPIG being non-negative as mutual information in two terms and thus so is JEPIG, which means the difference between the two BALD terms is non-negative and the second property is obtained from that.

Hence, as a tractable approximation $\hat{\boldsymbol{\Omega}}$, we choose to use a form of *self-distillation*, where we train a model with $\mathcal{D}^{\text{train}}$ and the predictions of the original model $\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$ on $\boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}$ using a KL-divergence loss, inspired by Hinton et al. [2015] and Zhang et al. [2019b][1]. The loss function this is:

$$\begin{aligned} &L(\boldsymbol{x}_{1..\mathtt{N}}^{\text{train}}, \mathrm{p}(\boldsymbol{\omega}), \mathrm{q}(\hat{w} \mid \mathcal{D}^{\text{train}})) = \\ &\quad = \frac{1}{|\mathcal{D}^{\text{train}}|} \sum_i \mathrm{D}_{\mathrm{KL}}(\mathrm{p}(Y \mid \boldsymbol{x}_i^{\text{train}}, \mathcal{D}^{\text{train}}) \parallel \mathrm{q}(Y \mid \boldsymbol{x}_i^{\text{train}})) \\ &\qquad + \mathrm{D}_{\mathrm{KL}}(\mathrm{q}(\boldsymbol{\omega}) \parallel \mathrm{p}(\boldsymbol{\omega})), \end{aligned} \tag{F.47}$$

with $\mathrm{p}(Y \mid \boldsymbol{x}_i^{\text{train}}) = \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})}\, \mathrm{p}(Y \mid \boldsymbol{x}_i^{\text{train}}, \boldsymbol{\omega})$ and $\mathrm{q}(Y \mid \boldsymbol{x}_i^{\text{train}}) = \mathbb{E}_{\mathrm{q}(\hat{\boldsymbol{\omega}})}\, \mathrm{p}(Y \mid \boldsymbol{x}_i^{\text{train}}, \hat{\boldsymbol{\omega}})$. The resulting model posterior $\hat{\boldsymbol{\Omega}}$ fulfills both properties described above. This is similar to self-distillation in that we train a new model on the predictions of the original model. However, self-distillation does not use predictions on otherwise unlabeled data. It is also similar to semi-supervised learning [Lee et al., 2013; Yarowsky, 1995] in that we use the predictions of the model on unlabeled data to train a new model. However, semi-supervised learning only uses the samples for which the model is most confident via confidence thresholding and either temperature-scales them for training (soft pseudo-labels) or takes the argmax (hard pseudo-labels) whereas we use the predictions without change for all evaluation samples.

Advantages of JEPIG. Compared to JEPIG, when evaluating

$$\mathrm{I}[Y_{1..\mathtt{E}}^{\text{eval}}; Y_{1..\mathtt{K}}^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}}] \tag{F.48}$$
$$= \mathrm{H}[Y_{1..\mathtt{K}}^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, \mathcal{D}^{\text{train}}] \tag{F.49}$$
$$\quad - \mathrm{H}[Y_{1..\mathtt{K}}^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}, Y_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}}]$$
$$\approx \mathrm{H}_{\mathrm{q}_\emptyset}[Y_{1..\mathtt{K}}^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}] - \frac{1}{M} \sum_i \mathrm{H}_{\mathrm{q}_i}[Y_{1..\mathtt{K}}^{\text{acq}} \mid \boldsymbol{x}_{1..\mathtt{K}}^{\text{acq}}] \tag{F.50}$$

with separate approximate Bayesian models $\mathrm{q}_\emptyset$ and q_i where $\mathrm{q}_\emptyset(\boldsymbol{\omega}) \approx \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}^{\text{train}})$ and $\mathrm{q}_i(\boldsymbol{\omega}) \approx \mathrm{p}(\boldsymbol{\omega} \mid {y_{1..n}}^i, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}})$ for ${y_{1..n}}^i \sim \mathrm{p}(y_{1..\mathtt{E}}^{\text{eval}} \mid \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}}), i \in 1..M$ for M draws of pseudo-labels for $y_{1..\mathtt{E}}^{\text{eval}}$, we found that:

$$\mathrm{H}(\mathrm{q}_\emptyset(Y \mid x, \boldsymbol{\Omega})) \neq \frac{1}{M} \sum_i \mathrm{H}(\mathrm{q}_i(Y \mid x, \boldsymbol{\Omega})),$$

[1] Essentially using $\alpha = 1, \lambda = 0$

which violates the modelling assumption

$$\mathrm{H}[Y \mid x, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}] = \mathrm{H}[Y \mid x, \boldsymbol{\Omega}, Y_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \mathcal{D}^{\text{train}}]$$

as $Y \perp\!\!\!\perp Y_{1..\mathtt{E}}^{\text{eval}} \,|\, \boldsymbol{x}, \boldsymbol{x}_{1..\mathtt{E}}^{\text{eval}}, \boldsymbol{\Omega}$. This is even more of an issue when using a single approximate model with the self-distillation described in the previous section because the two properties we wish for will force $\mathrm{H}[Y \mid x, \hat{\boldsymbol{\Omega}}] \neq \mathrm{H}[Y \mid x, \mathcal{D}^{\text{train}}]$. This follows immediately from the expansion of the mutual information: $\mathrm{I}[Y; \boldsymbol{\Omega} \,|\, x, \mathcal{D}^{\text{train}}] = \mathrm{H}[Y \,|\, x, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y \,|\, x, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}]$ as the first property will fix $\mathrm{H}[Y \mid x, \mathcal{D}^{\text{train}}]$ and the second will force the $\mathrm{H}[Y \mid x, \boldsymbol{\Omega}, \mathcal{D}^{\text{train}}]$ terms apart to achieve the inequality. However, JEPIG does not need this assumption as it explicitly estimates the epistemic uncertainty, and thus performs better when using our approximation with self-distillation. Moreover, it has its own strong intuitive motivation.

G

Prioritized Data Selection during Training

G.1 Steps Required for a Given Test Accuracy

Figs. G.1 (vision) and G.2 (NLP) show the number of steps required to reach a given test accuracy across several datasets for different selection methods. Interestingly, on CoLA (unbalanced and noisy), the uniform sampling baseline shows high variance across seeds, while RHO-LOSS works robustly across seeds.

Table G.1 shows results for RHO-LOSS training without holdout data. Results are similar to Table 8.2. Here, we train the IL model without any holdout data. We split the training set $\mathcal{D}^{\text{pool}}$ into two halves and train an IL model on each half. Each model computes the IL for the half of $\mathcal{D}^{\text{pool}}$ that it was not trained on. (This is as in Figure 8.2, row 3, except that previously we only used half of $\mathcal{D}^{\text{pool}}$ and further split it into halves of the half.) Training two IL models costs no additional compute since each model is trained on half as much data compared to the default settings.

G.2 Experiment Details

Architectures. We experiment with various architectures in Figs. 8.1 and 8.2 (row 4). In all other figures and tables, we use the following architectures: For experiments on QMNIST, we use a multi-layer perceptron with 2 hidden layers and 512 units in each hidden layer. For experiments on CIFAR-10, CIFAR-100 and CINIC-10, we use a variant of ResNet-18 [He et al., 2016]. We adapted the ResNet18 to 32x32 images by modifying the architecture to remove the downsampling effect. We replaced the spatial downsampling of a strided convolution and max pooling in the original ResNet18, with a convolutional layer with 64 filters and a kernel size of 3x3. We also removed the average pooling at the end of the ResNet18. This ResNet18 variant is similar to Resnet20, just with more filters. For experiments on Clothing-1M, following the experimental set-up of Yi and Wu [2019], the target model is a ResNet-50 pre-trained on ImageNet. The irreducible loss model is a ResNet-18 with random initialization. The multiple target architectures in Fig 8.2 were adapted from **?**. For NLP datasets, we use a pre-trained ALBERT v2 [Lan et al., 2020].

Hyperparameters. *Vision*: All models are trained using the AdamW optimizer with default PyTorch hyperparameters (β_1=0.9, β_2=0.999, and weight decay of 0.01, learning rate 0.001), a $\mathtt{K} = 32$ (64 for CINIC-10) $\mathtt{K}' = 320$ (640 for CINIC-10), meaning we select $\frac{\mathtt{K}}{\mathtt{K}'} = 10\%$ of points. *NLP*: ALBERT v2 was trained using the AdamW optimizer with a learning rate as indicated in the original paper ($2 \cdot 10^{-5}$) and weight decay of 0.02. We fine-tuned all weights, not just the final layer. The batch size $\mathtt{K}$ was 32, $\mathtt{K}' = 320$, meaning we select $\frac{\mathtt{K}}{\mathtt{K}'} = 10\%$ of points. We use between 2 and 10 seeds for each experiment.

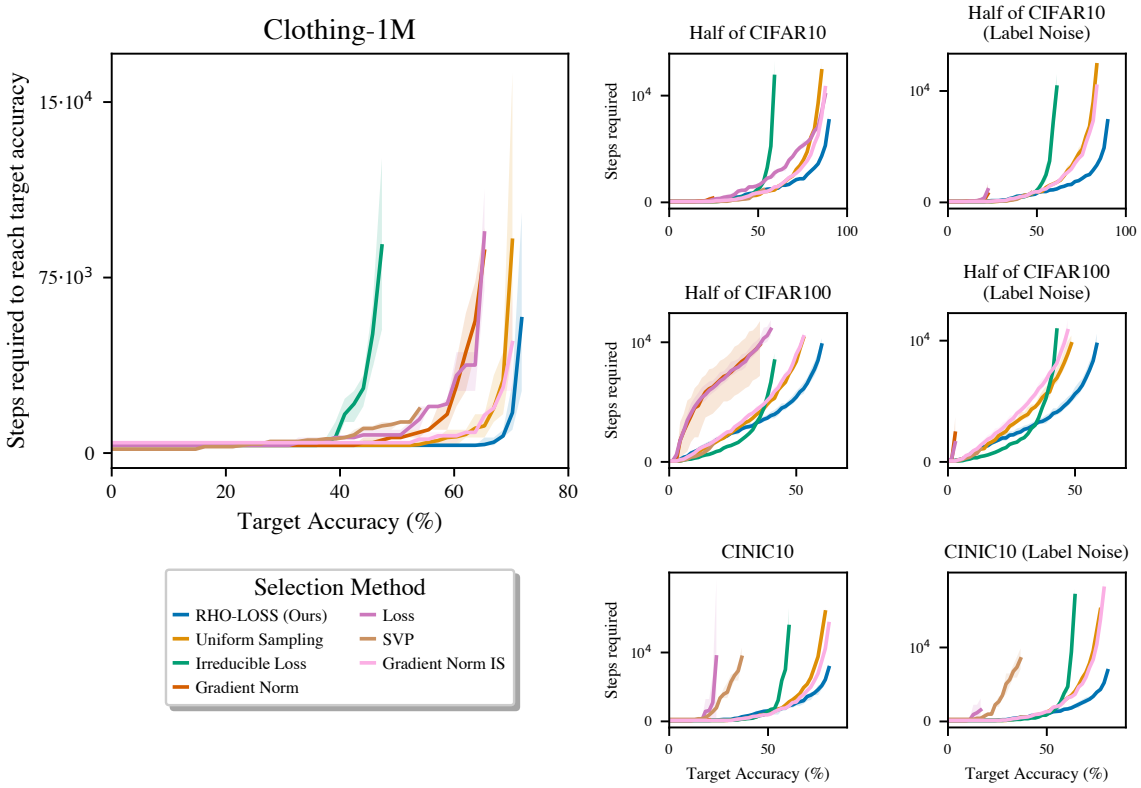


Figure G.1: Vision datasets—gradient steps required to achieve a given test accuracy **(lower is better)**. **Left column:** The speedup of RHO-LOSS over uniform sampling is the greatest on a large-scale web-scraped dataset with noisy labels. **Middle column:** Speedups are still substantial on clean datasets and RHO-LOSS still achieves higher final accuracy than all prior art. **Right column:** Applying 10% uniform label noise to training data degrades other methods but increases the speedup of our method. A step corresponds to lines $5-10$ in Algorithm 1. Lines correspond to means and shaded areas to minima and maxima across 3 random seeds. On CIFAR10/100, only half of the data is used for training (see text).

Data Augmentation. On CIFAR-10, CIFAR-100, and CINIC-10, we train using data augmentation (random crop and horizontal flip), both for training the IL model, and in the main training runs. Remember that we only compute the irreducible losses once at the start of training, to save compute (Algorithm 3). We use the unaugmented images for this as we found that using augmented images makes little difference to performance but costs more compute.

Irreducible Loss Model Training. The irreducible loss models are trained on holdout sets, i.e. labeled evaluation sets (not test sets, see dataset description in main text). For each dataset, we select the irreducible loss model checkpoint from the epoch with the *lowest holdout loss* on $\mathcal{D}^{\text{pool}}$ (as opposed to the highest accuracy); we find that this improves performance while also saving compute as the holdout loss typically reaches its minimum early in training.

BatchNorm. Like many deep-learning methods, RHO-LOSS interacts with BatchNorm Ioffe and Szegedy [2015] since the loss of a given point is affected by other points in the same batch. **Important:** We compute the BatchNorm statistics for selection and model update separately. For selection (line 5-8 in Algorithm 3), the statistics are

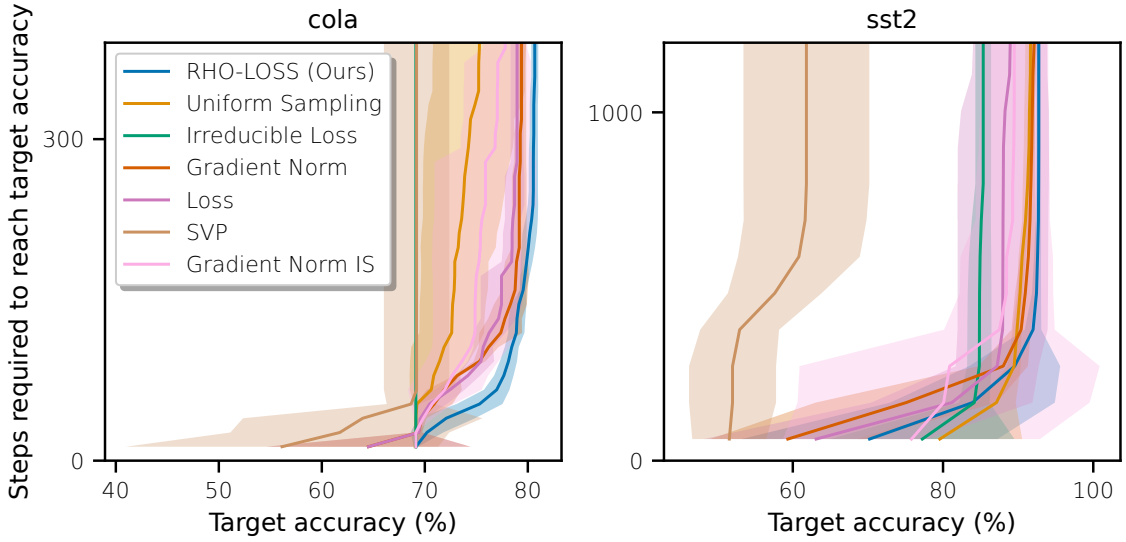


Figure G.2: NLP datasets—gradient steps required to achieve a given test accuracy **(lower is better)**. **Left:** CoLA grammatical acceptability classification. **Right:** SST2 sentiment classification. A step corresponds to lines $5-10$ in Algorithm 1. Lines correspond to means and shaded areas to standard deviations across 4 or more random seeds. Only half of the data is used for training (see text).

Table G.1: Epochs required to reach a given target test accuracy when using no holdout data (lower is better). Final accuracy in parentheses. Results averaged across 2-3 seeds. Best performance in bold. RHO-LOSS performs best in both epochs required and final accuracy.

Dataset	Target Accuracy	Uniform Sampling	RHO-LOSS
CIFAR10	80%	39	**17**
	90%	177 (90.8%)	**47 (92.2%)**
CIFAR100	50%	47	**22**
	65%	142 (67.8%)	**87 (68.1%)**
CINIC10	70%	37	**26**
	80%	146 (80.1%)	**70 (82.1%)**

computed across the large batch B_t. For training (line 9-10), the statistics are computed across the small batch b_t. These choices can affect performance a lot. For new datasets, we recommend varying how the batch-norm statistics are computed during selection (trying both train mode and evaluation mode) and choose the option that works best.

G.3 Robustness to Noise

In this set of experiments, we evaluate the performance of different selection methods under a variety of noise patterns on QMNIST (MNIST with extra holdout data) and variations thereof. We use this dataset because it has little label noise in its original form, allowing us to test the effect of adding noise. Firstly, we add uniform label noise to 10% of training points. Secondly, we add structured label noise that affects easily confused classes. We follow Rolnick et al. [2017] and flip the labels of the four most frequently confused classes (in the confusion matrix of a trained model) with 50%

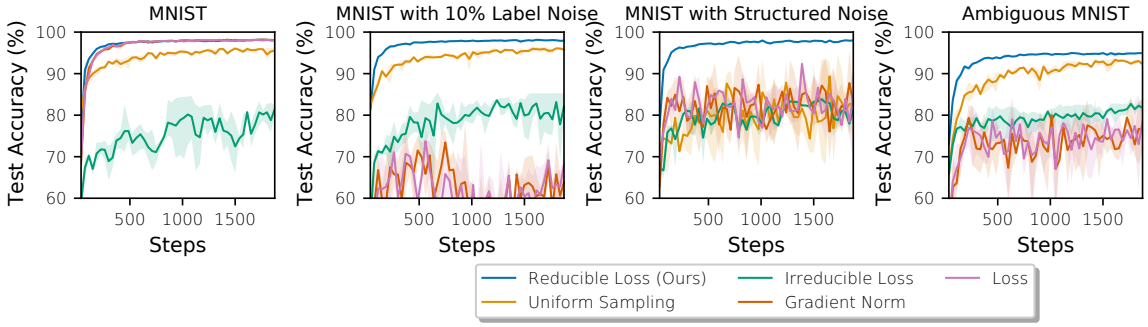


Figure G.3: RHO-LOSS is robust to a variety of label noise patterns, while other selection methods degrade. A step corresponds to lines $6-11$ in Algorithm 3. Lines correspond to means and shaded areas to minima and maxima across 3 random seeds.

probability. For example, a 2 is often confused with a 5; thus we change the label of all 2s to 5s with 50% probability. Thirdly, we leverage the natural noise distribution of MNIST by using Ambiguous-MNIST [Mukhoti et al., 2023] as the training set. Ambiguous-MNIST contains a training set with 60k generated ambiguous digits that have more than one plausible label. While selecting with loss and gradient norm trains accelerates training on the MNIST training set, their performance degrades on all three types of noise distributions (Figure G.3).

G.4 Irreducible Holdout Loss Approximation

In this appendix section, we examine one of the key approximations made in the theory section. To arrive at Eq. (8.11), we used the approximation $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}] \approx \mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}, \mathcal{D}^{\text{train}}]$. In words, we approximated the cross-entropy loss of a model trained on the data points acquired so far $\mathcal{D}^{\text{train}}$ and the holdout dataset $\mathcal{D}^{\text{eval}}$, with the cross-entropy loss of a model trained only on the holdout set (i.e. the *labeled* evaluation set). This approximation saves a lot of compute: rather than having to recompute the term with every change of $\mathcal{D}_t$, it is now sufficient to compute it once at the start of training.

We have already highlighted the impact of the approximation on points selected when training on QMNIST in §8.3.1. In our main experiment setting—using neural networks trained with gradient descent—we empirically find that the approximation does not reduce speed of target model training or final target model accuracy (Table G.2). This finding holds across a range of datasets (CIFAR-10, CIFAR-100, CINIC-10). Updating the irreducible loss model on $\mathcal{D}^{\text{train}}$ seems empirically not necessary.

Indeed, the approximation actually has two desirable properties when used for neural networks trained with gradient descent. We will first describe why we expect these desirable properties, and then show that they indeed appear. First, let us restate both selection functions:

- JPIG:

$$\underset{(x,y)\in B_t}{\arg\max}\ \mathrm{H}[y \mid x, \mathcal{D}^{\text{train}}] - \mathrm{H}[y \mid x, \mathcal{D}^{\text{eval}}, \mathcal{D}^{\text{train}}]; \textit{and} \tag{G.1}$$

- approximated JPIG:

$$\underset{(x,y)\in B_t}{\arg\max}\ \mathrm{H}[y \mid x, \mathcal{D}^{\text{train}}] - \mathrm{H}[y \mid x, \mathcal{D}^{\text{eval}}]. \tag{G.2}$$

Table G.2: Number of epochs required to reach a given target test accuracy across several datasets. Results averaged across 2-3 random seeds. NR indicates that the target accuracy was not reached.

Dataset	Target accuracy	RHO-LOSS: Approximated JPIG $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] - \mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}]$	JPIG $\mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{train}}] - \mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}^{\text{eval}}, \mathcal{D}^{\text{train}}]$
CIFAR10	60%	18	13
	75%	30	24
	90%	102	NR, but reaches 88% in 157 epochs
CIFAR100	30%	35	21
	45%	58	NR, but reaches 43% in 61 epochs
	60%	123	NR
CINIC10	55%	12	12
	65%	19	21
	75%	32	NR, but reaches 74% in 68 epochs

Desirable property 1. *The approximation prevents repeated selection of undesirable points.* When using SGD instead of Bayesian updating, the original selection function can acquire undesired points repeatedly. Let's say that we acquire, for whatever reason, a noisy, redundant, or irrelevant point. We only take one gradient step each time we acquire a (batch of) point(s), meaning the training loss (first term in the selection function) will on each only decrease somewhat. In the original selection function, the second term will also decrease somewhat, meaning that the difference between the first and second term may remain large. In the approximated selection function, the second term is constant, the difference between first and second term will thus likely decrease more than under the original selection function. Under the approximated selection function, we are thus less likely to acquire undesired points again, if we have acquired them in earlier epochs.

Desirable property 2. *The approximation prevents deterioration of the irreducible loss model over time.* With both selection functions, we compute the second term of the selection function with an "irreducible loss model", which we train on a holdout set (i.e. a *labeled* evaluation set) before we start target model training. In the target model training, we (greedily) acquire the points that most improve the loss of the target model (on the holdout set, the *labeled* evaluation set). We thus deliberately introduce bias into the data selection. However, this bias is tailored to the target model and may not be suitable for the irreducible loss model. As a simplifying example, consider a target model early in training, which has not yet learned a certain class, and an irreducible loss model, which has learned that class. Data points in that class will have high training loss, low irreducible loss, and will be acquired often. This, however, is not useful for the irreducible loss model, and might lead to decreased accuracy on data points from other classes. With the approximation, this can't happen. The described failure mode could likely also be alleviated by more sophisticated training schemes for the irreducible loss model, such as periodically mixing in data points from the holdout set (i.e. the *labeled* evaluation set). However, such training schemes would require even more compute and/or overhead.

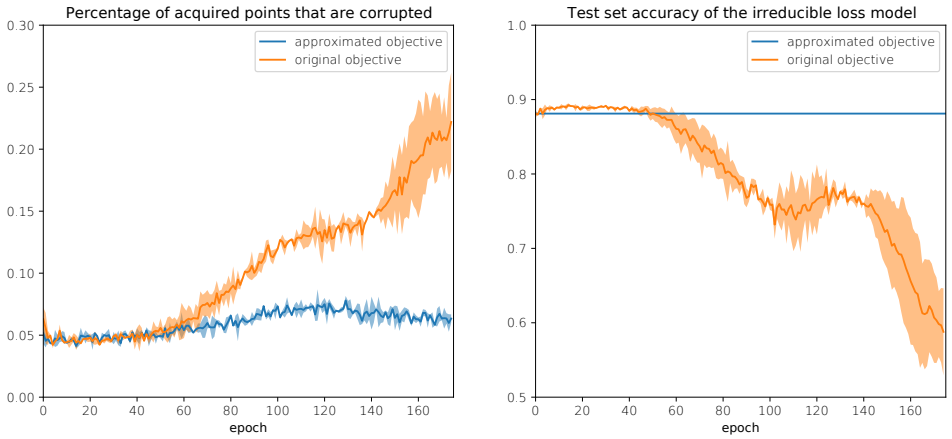


Figure G.4: Desired properties of the irreducible loss model approximation. **Left.** The approximated selection function selects fewer corrupted points later on in training. **Right.** The test set accuracy of the irreducible loss model deteriorates over time if it is updated on $\mathcal{D}_t$. With the approximation, the irreducible loss is not updated during target model training. Results on CIFAR-10 with 20% of data points corrupted with uniform label noise. Shaded areas represent standard deviation across three different random seeds.

We find empirically that both desired properties of the approximation indeed manifest themselves. In Figure G.4, we train a target model (Resnet-18) on CIFAR-10, with 20% of the data points corrupted by uniform label noise. The approximated selection function leads to faster target model training (the approximated selection function needs 80 epochs to reach the same target model accuracy that the original selection function reaches in 100 epochs) and higher final accuracy than the original selection function (88.6% vs 86.1%). Indeed, the original selection function leads to acquiring more corrupted points, especially later in training (Figure G.4, left), and the accuracy *of the irreducible loss model* deteriorates over time (Figure G.4, right). We tuned the learning rate of the irreducible loss model to 0.01 times that of the target model. Without this adjustment, the results look similar but the original selection function performs worse.

G.5 Experimental Details for Assessing Impact of Approximations

Dataset. QMNIST, with uniform label noise applied to 10% of the dataset. Batch size of 1000 is used.

Models. Deep Ensemble contains 5 3-layer MLP's with 512 hidden units. The weaker irreducible loss model is an MLP with 256 hidden units.

Training. For Approximation 0, we use a deep ensemble for both models. The irreducible loss model is trained to convergence on $\mathcal{D}^{\text{eval}}$. Then the target model and the irreducible model are used to acquire 10% of points each batch using the selection function. They are then trained to convergence on each batch of points acquired. The irreducible loss model is trained on $\mathcal{D}^{\text{eval}} \cup \mathcal{D}^{\text{train}}$, while the target model is only trained on $\mathcal{D}^{\text{train}}$. We train for a maximum of 5 epochs, which often is to convergence, to

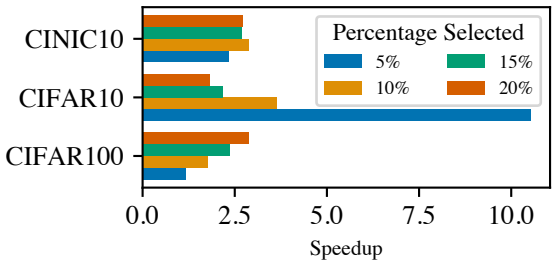


Figure G.5: Varying the percent of data points selected in each training batch. Average over 3 random seeds.

enable a fair comparison to further approximations. For Approximation 1a, the deep ensembles are replaced with single MLPs. The training regime remains the same. We compare the approximations over the first epoch. To compare Approximation 1b to 0, and for all further approximations, we increase the size of the dataset five-fold, by duplicating samples in QMNIST. This means for approximation 1b, we have 5x the data that we have for Approximation 1a, but with increased redundancy. We train the model in Approximation 1b by taking a single gradient step per data point, with the larger dataset. On the other hand, we train the model for Approximation 0 (still to convergence or 5 epochs) on the standard dataset size. By doing this, Approximation 0 and 1b have taken the equivalent number of gradient steps, at the time-steps where we are tracking the reducible loss of points selected, enabling a fair comparison between the approximations. The irreducible loss models are trained on $\mathcal{D}^{\text{eval}} \cup \mathcal{D}^{\text{train}}$ in their respective set-ups. To compare Approximation 2 to Approximation 0, we compare updating the irreducible loss model with a single gradient on each set of acquired points, to not updating the irreducible loss model on $\mathcal{D}^{\text{train}}$ at all. To isolate effect of not updating, we utilize the same initial irreducible loss model. To compare Approximation 3, we simply train a small irreducible model (one with 256 hidden units) and follow the same training regime as Approximation 2.

G.6 Ablation of Percentage Selected

Our method has a hyperparameter, the percentage $\frac{\mathtt{K}}{\mathtt{K}'}$ of evaluated points which are selected for training. In the experiments above, this parameter was set to 0.1. We have not tuned this parameter, as we aim to analyze how well our method works "out of the box". In fact, on 2/3 datasets, performance further improves with other values of this parameter. Adjusting this percentage should allow practitioners to specify their preferred trade-off between training time and computation, where a low percentage typically corresponds to a lower training time and greater compute cost. For these experiments, we kept $\mathtt{K} = 32$ and adapt $\mathtt{K}'$ accordingly. The percentage $\frac{\mathtt{K}}{\mathtt{K}'}$ of data points selected per batch has different effects across datasets as shown in Figure G.5.

G.7 Active Learning Baselines

We compare our method to typical methods used in the Active Learning (AL) literature. Note that our method is label-aware, while active learning acquires data points without using labeled information. We consider the following baselines, which select the top-k

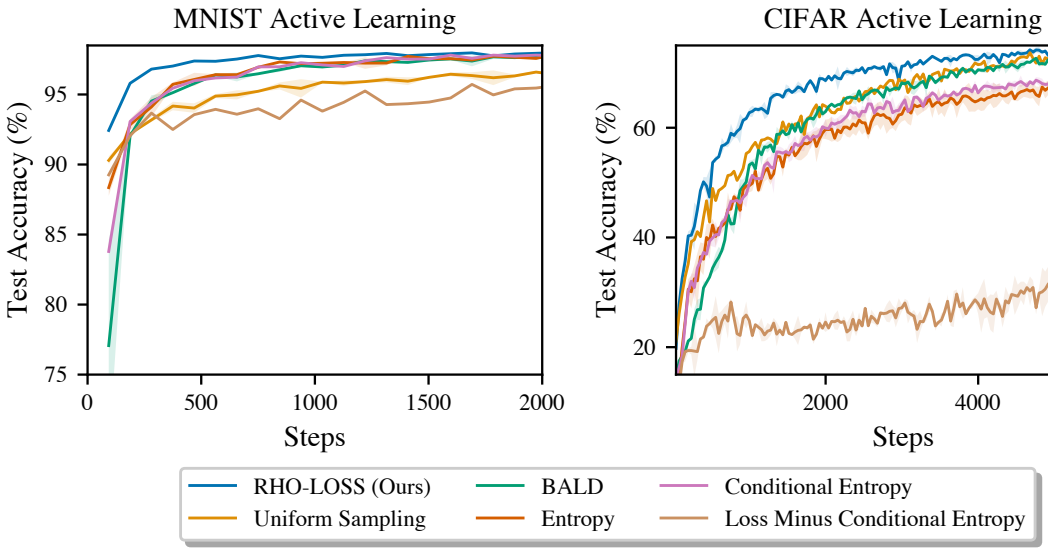


Figure G.6: Training curves for several active learning baselines on the MNIST and CIFAR10 datasets.

points using an acquisition function, $\alpha(\boldsymbol{x})$:

- Bayesian Active Learning by Disagreement [Houlsby et al., 2011] with $\alpha(\boldsymbol{x}) = \mathrm{H}[y|\boldsymbol{x}, \mathcal{D}_t] - \mathbb{E}_{\mathrm{p}(\theta|\mathcal{D}_t)}[\mathrm{H}[y \mid \boldsymbol{x}, \theta]]$.
- (Average) conditional entropy, $\alpha(\boldsymbol{x}) = \mathbb{E}_{\mathrm{p}(\theta|\mathcal{D}_t)}[\mathrm{H}[y \mid \boldsymbol{x}, \theta]]$, where the average is taken over the model parameter posterior.
- (Average predictive) entropy, $\alpha(\boldsymbol{x}) = \mathrm{H}[y \mid \boldsymbol{x}, \mathcal{D}_t]$.
- Loss minus conditional entropy $\alpha(\boldsymbol{x}) = \mathrm{H}[y \mid \boldsymbol{x}, \theta] - \mathbb{E}_{\mathrm{p}(\theta|\mathcal{D}_t)}[\mathrm{H}[y \mid \boldsymbol{x}, \theta]]$. This uses the (average) conditional entropy as an estimate of how noisy data point $\boldsymbol{x}$ is—points with high noise are deprioritized. Compared to RHO-LOSS, it replaces the IL with the conditional entropy. This acquisition function uses the label and therefore cannot be used for active learning.

We additionally compare our method to uniform sampling. We run all baselines on MNIST and CIFAR10. Note that several of these active learning baselines consider epistemic uncertainty; that is, uncertainty in predictions driven by uncertainty in the model parameters. This mandates performing (approximate) Bayesian inference. We use Monte-Carlo Dropout[Gal and Ghahramani, 2016a] to perform approximate inference. For MNIST, we use an 2 hidden layer MLP with 512 hidden units per hidden layer, and a dropout probability of a 0.5. For experiments on CIFAR10, we use a small-scale CNN with dropout probability 0.05 (the dropout probability follows [Osawa et al., 2019]).

Figure G.6 shows training curves for our method, uniform sampling, and the active learning baselines. Our method accelerates training across both datasets. The active learning methods accelerate training for MNIST but not for CIFAR10. This highlights that active learning methods, if naively applied to online batch selection, may not accelerate model training.

H

Unifying Approaches in Active Learning and Active Sampling

H.1 Fisher Information: Additional Derivations & Proofs

Proposition 9.6. *Like observed information, Fisher information is additive:*

$$\mathrm{H}''[\{Y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*] = \sum_i \mathrm{H}''[\{Y_i\} \mid x_i, \omega^*]. \tag{9.16}$$

Proof. This follows immediately from $Y_i \perp\!\!\!\perp Y_j \mid \boldsymbol{x}_i, \boldsymbol{x}_j, \omega^*$ for $i \neq j$ and the additivity of the observed information:

$$\mathrm{H}''[\{Y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*] = \mathbb{E}_{\mathrm{p}(\{y_i\}|\{\boldsymbol{x}_i\},\omega^*)}[\mathrm{H}''[\{y_i\} \mid \{\boldsymbol{x}_i\}, \omega^*]] = \mathbb{E}_{\mathrm{p}(\{y_i\}|\{\boldsymbol{x}_i\},\omega^*)}[\sum_i \mathrm{H}''[y_i \mid x_i, \omega^*]] \tag{H.1}$$

$$= \sum_i \mathbb{E}_{\mathrm{p}(y_i|x_i,\omega^*)}[\mathrm{H}''[y_i \mid x_i, \omega^*]] = \sum_i \mathrm{H}''[y_i \mid x_i, \omega^*]. \tag{H.2}$$

□

Proposition 9.7. *Fisher information is equivalent to:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\mathrm{H}'[y \mid x, \omega^*]^T \ \mathrm{H}'[y \mid x, \omega^*]] = \mathrm{Cov}[\mathrm{H}'[Y \mid x, \omega^*]]. \tag{9.17}$$

To prove Proposition 9.7, we use the two generally useful lemmas below:

Lemma H.1. *For the Jacobian* $\mathrm{H}'[y \mid \boldsymbol{x}, \omega^*]$*, we have:*

$$\mathrm{H}'[y \mid x, \omega^*] = \nabla_\omega[-\log \mathrm{p}(y \mid x, \omega^*)] = -\frac{\nabla_\omega \mathrm{p}(y \mid x, \omega^*)}{\mathrm{p}(y \mid x, \omega^*)}, \tag{H.3}$$

and for the Hessian $\mathrm{H}''[y \mid \boldsymbol{x}, \omega^*]$*, we have:*

$$\mathrm{H}''[y \mid x, \omega^*] = \mathrm{H}'[y \mid x, \omega^*]^T \ \mathrm{H}'[y \mid x, \omega^*] - \frac{\nabla^2_\omega \mathrm{p}(y \mid x, \omega^*)}{\mathrm{p}(y \mid x, \omega^*)}. \tag{H.4}$$

Proof. The result follows immediately from the application of the rules of multivariate calculus. □

Lemma H.2. *The following expectations over the model's own predictions vanish:*

$$\mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\mathrm{H}'[y \mid x, \omega^*]] = 0, \tag{H.5}$$

$$\mathbb{E}_{\mathrm{p}(y|x,\omega^*)}\left[\frac{\nabla^2_\omega \,\mathrm{p}(y \mid x, \omega^*)}{\mathrm{p}(y \mid x, \omega^*)}\right] = 0. \tag{H.6}$$

Proof. We use the previous equivalences and rewrite the expectations as integral; the results follows:

$$\mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\mathrm{H}'[y \mid x, \omega^*]] = \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[-\nabla_\omega \log \mathrm{p}(y \mid x, \omega^*)] \tag{H.7}$$

$$= -\,\mathbb{E}_{\mathrm{p}(y|x,\omega^*)}\left[\frac{\nabla_\omega \,\mathrm{p}(y \mid x, \omega^*)}{\mathrm{p}(y \mid x, \omega^*)}\right]$$

$$= -\int \nabla_\omega \,\mathrm{p}(y \mid x, \omega^*)\, dy = -\nabla_\omega \int \mathrm{p}(y \mid x, \omega^*)\, dy = -\nabla_\omega 1 = 0, \tag{H.8}$$

$$\mathbb{E}_{\mathrm{p}(y|x,\omega^*)}\left[\frac{\nabla^2_\omega \,\mathrm{p}(y \mid x, \omega^*)}{\mathrm{p}(y \mid x, \omega^*)}\right] = \int \nabla^2_\omega \,\mathrm{p}(y \mid x, \omega^*)\, dy \tag{H.9}$$

$$= \nabla^2_\omega \int \mathrm{p}(y \mid x, \omega^*)\, dy \tag{H.10}$$

$$= \nabla^2_\omega 1 = 0. \tag{H.11}$$

□

Proof of Proposition 9.7. With the previous lemma, we have the following.

$$\mathrm{Cov}[\mathrm{H}'[Y \mid x, \omega^*]] = \mathbb{E}[\mathrm{H}'[Y \mid x, \omega^*]^T\ \mathrm{H}'[Y \mid x, \omega^*]]x \tag{H.12}$$

$$- \underbrace{\mathbb{E}[\mathrm{H}'[Y \mid x, \omega^*]^T]}_{=0}\ \underbrace{\mathbb{E}[\mathrm{H}'[Y \mid x, \omega^*]]}_{=0}$$

$$= \mathbb{E}[\mathrm{H}'[Y \mid x, \omega^*]^T\ \mathrm{H}'[Y \mid x, \omega^*]]. \tag{H.13}$$

For the expectation over the Hessian, we plug Lemma H.1 into Lemma H.2 and obtain:

$$\mathrm{H}''[Y \mid x, \omega^*] = \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\mathrm{H}''[y \mid x, \omega^*]] \tag{H.14}$$

$$= \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}\left[\mathrm{H}'[y \mid x, \omega^*]^T\ \mathrm{H}'[y \mid x, \omega^*] - \frac{\nabla^2_\omega \,\mathrm{p}(y \mid x, \omega^*)}{\mathrm{p}(y \mid x, \omega^*)}\right] \tag{H.15}$$

$$= \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\mathrm{H}'[y \mid x, \omega^*]^T\ \mathrm{H}'[y \mid x, \omega^*]] - 0 \tag{H.16}$$

$$= \mathrm{Cov}[\mathrm{H}'[Y \mid x, \omega^*]]. \tag{H.17}$$

□

H.1.1 Special Case: Exponential Family

Proposition 9.8. *The Fisher information* $\mathrm{H}''[Y \mid \boldsymbol{x}, \omega^*]$ *for a model* $\mathrm{p}(y \mid \hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}; \omega^*))$ *is equivalent to:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \nabla_\omega \hat{f}(x; \omega^*)^T\, \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\nabla^2_{\hat{\boldsymbol{z}}}\, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x; \omega^*)]] \nabla_\omega \hat{f}(x; \omega^*), \tag{9.18}$$

where $\nabla^2_{\hat{\boldsymbol{z}}}\, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}; \omega^*)]$ *is short for* $\nabla^2_{\hat{\boldsymbol{z}}}\, \mathrm{H}[y \mid \hat{\boldsymbol{z}}]|_{\hat{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}; \omega^*)}$.

Proof. We apply the second equivalence in Proposition 9.7 twice:

$$\begin{aligned}
\mathrm{H}''[Y \mid x, \omega^*] &= \mathrm{Cov}[\mathrm{H}'[Y \mid x, \omega^*]] && \text{(H.18)}\\
&= \mathrm{Cov}[\nabla_\omega \hat{f}(x;\omega^*)^T \, \nabla_{\hat{z}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x;\omega^*)] \, \nabla_\omega \hat{f}(x;\omega^*)] && \text{(H.19)}\\
&= \nabla_\omega \hat{f}(x;\omega^*)^T \, \mathrm{Cov}[\nabla_{\hat{z}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x;\omega^*)]] \, \nabla_\omega \hat{f}(x;\omega^*) && \text{(H.20)}\\
&= \nabla_\omega \hat{f}(x;\omega^*)^T \, \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\nabla^2_{\hat{z}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x;\omega^*)]] \, \nabla_\omega \hat{f}(x;\omega^*) && \text{(H.21)}
\end{aligned}$$

□

H.1.2 Special Case: Generalized Linear Models

Proposition 9.10. *The observed information* $\mathrm{H}''[y \mid \boldsymbol{x}, \omega^*]$ *of a GLM is independent of* y*.*

$$\begin{aligned}
\mathrm{H}''[y \mid x, \omega^*] &= \nabla_\omega \hat{f}(x;\omega^*)^T \, \nabla^2_{\hat{z}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x;\omega^*)] \, \nabla_\omega \hat{f}(x;\omega^*) && \text{(9.22)}\\
&= \nabla_\omega \hat{f}(x;\omega^*)^T \, \nabla^2_{\hat{z}} A(w^T x) \, \nabla_\omega \hat{f}(x;\omega^*). && \text{(9.23)}
\end{aligned}$$

Proof.

$$\begin{aligned}
&\mathrm{H}''[y \mid x, \omega^*] && \text{(H.22)}\\
&\quad = \nabla_\omega [\mathrm{H}'[y \mid x, \omega^*]] && \text{(H.23)}\\
&\quad = \nabla_\omega [\nabla_{\hat{z}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x;\omega^*)] \, \nabla_\omega \hat{f}(x;\omega^*)] && \text{(H.24)}\\
&\quad = \nabla_{\hat{z}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x;\omega^*)] \, \underbrace{\nabla^2_\omega \hat{f}(x;\omega^*)}_{=\nabla^2_\omega [w^T x] = 0} && \text{(H.25)}\\
&\qquad + \nabla_\omega \hat{f}(x;\omega^*)^T \, \nabla^2_{\hat{z}} \, \mathrm{H}[y \mid \hat{\boldsymbol{z}} = \hat{f}(x;\omega^*)] \, \nabla_\omega \hat{f}(x;\omega^*)\\
&\quad = \nabla_\omega \hat{f}(x;\omega^*)^T \, \nabla^2_{\hat{z}} A(w^T x) \, \nabla_\omega \hat{f}(x;\omega^*). && \text{(H.26)}
\end{aligned}$$

□

Proposition 9.12. *For a GLM, when* $\hat{f}(\boldsymbol{x};\omega) : \mathbb{R}^D \to \mathbb{R}^{\mathtt{C}}$*, where* $\mathtt{C}$ *is the number of classes (outputs),* D *is the number of input dimensions,* $\omega \in \mathbb{R}^{D \times \mathtt{C}}$*, and assuming the parameters are flattened into a single vector for the Jacobian, we have* $\nabla_\omega \hat{f}(\boldsymbol{x};\omega) = \mathrm{Id}_{\mathtt{C}} \otimes \boldsymbol{x}^T \in \mathbb{R}^{\mathtt{C} \times (\mathtt{C} \cdot D)}$*, where* $\otimes$ *denotes the Kronecker product, and:*

$$\nabla_\omega \hat{f}(x;\omega^*)^T \, \nabla^2_{\hat{z}} A(\omega^T x) \, \nabla_\omega \hat{f}(x;\omega^*) = \nabla^2_{\hat{z}} A(w^T x) \otimes x \, x^T. \tag{9.26}$$

Proof. We begin with a few statements that lead to the conclusion step by step, where $\boldsymbol{x} \in \mathbb{R}^D, A \in \mathbb{R}^{C \times C}, G \in \mathbb{R}^{C \times (C \cdot D)}$:

$$\begin{aligned}
(x \, x^T)_{ij} &= x_i \, x_j && \text{(H.27)}\\
(Id_C \otimes x^T)_{c, d\,D+i} &= x_i \cdot \mathbb{1}\{c = d\} && \text{(H.28)}\\
(G^T \, A \, G)_{ij} &= \sum_{k,l} G_{ki} \, A_{kl} \, G_{lj} && \text{(H.29)}\\
(A \otimes x \, x^T)_{c\,D+i, d\,D+j} &= A_{cd} \, x_i \, x_j, && \text{(H.30)}\\
((Id_C \otimes x^T)^T \, A \, (Id_C \otimes x^T))_{c\,D+i, d\,D+j} &= \sum_{k,l} (Id_C \otimes x^T)_{k, c\,D+i} \, A_{kl} \, (Id_C \otimes x^T)_{l, d\,D+j} && \text{(H.31)}
\end{aligned}$$

$$= \sum_{k,l} x_i \cdot \mathbb{1}\{k = c\}\, A_{kl}\, x_j \cdot \mathbb{1}\{l = d\} \tag{H.32}$$

$$= x_i\, A_{cd}\, x_j \tag{H.33}$$

$$= (A \otimes x\, x^T)_{c\,D+i,d\,D+j}. \tag{H.34}$$

$$\implies \nabla_\omega \hat{f}(x;\omega^*)^T\, \nabla^2_{\hat{z}} A(\omega^T x)\, \nabla_\omega \hat{f}(x;\omega^*) = \nabla^2_{\hat{z}} A(w^T x) \otimes x\, x^T. \tag{H.35}$$

□

Proposition 9.11. *For a model such that the observed information* $\mathrm{H}''[y \mid \boldsymbol{x}, \omega^*]$ *is independent of* y*, we have:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \mathrm{H}''[y^* \mid x, \omega^*] \tag{9.24}$$

for any y^**, and also trivially:*

$$\mathrm{H}''[Y \mid x, \omega^*] = \mathbb{E}_{\mathrm{p}(y|x)}[\mathrm{H}''[y \mid x, \omega^*]]. \tag{9.25}$$

Proof. This follows directly from Proposition 9.9. In particular, we have:

$$\mathrm{H}''[Y \mid x, \omega^*] = \mathbb{E}_{\mathrm{p}(y|x,\omega^*)}[\mathrm{H}''[y \mid x, \omega^*]] = \mathrm{H}''[y^* \mid x, \omega^*], \tag{H.36}$$

where we have fixed y^* to an arbitrary value. □

H.2 Approximating Information Quantities

H.2.1 Approximate Expected Information Gain

Lemma 9.13. *For symmetric, positive semi-definite matrices* A*, we have (with equality iff* $A = 0$*):*

$$\log\det(A + Id) \leq \mathrm{tr}(A). \tag{9.38}$$

Proof. When A is positive semi-definite and symmetric, its eigenvalues $(\lambda_i)_i$ are real and non-negative. Moreover, $A + Id$ has eigenvalues $(\lambda_i + 1)_i$; $\det(A + Id) = \prod_i (\lambda_i + 1)$; and $\mathrm{tr}\, A = \sum_i \lambda_i$. These properties easily follow from the respective eigenvalue decomposition. Thus, we have:

$$\log\det(A + Id) \leq \log \prod_i (\lambda_i + 1) = \sum_i \log(\lambda_i + 1) \leq \sum_i \lambda_i = \mathrm{tr}(A), \tag{H.37}$$

where we have used $\log(x + 1) \leq \boldsymbol{x}$ iff equality for $\boldsymbol{x} = 0$. □

General Case. In the main text, we only skimmed the general case and mentioned the main assumption. Here, we look at the general case in detail.

For the general case, we need to make strong approximations to be able to pursue a similar derivation. First, we cannot drop the expectation; instead, we note that the log determinant is a concave function on the positive semi-definite symmetric cone [Cover and Thomas, 1988], and we can use Jensen's inequality on the log determinant term from Equation 9.32 as follows:

$$\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[\log\det\left(\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]\ \mathrm{H}''[\omega^*]^{-1} + Id\right)] \tag{H.38}$$

$$\leq \log\det\left(\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]]\ \mathrm{H}''[\omega^*]^{-1} + Id\right). \tag{H.39}$$

Second, we need to use the following approximation:

$$\mathrm{p}(\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}) \approx \mathrm{p}(\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*) \tag{H.40}$$

to obtain a Fisher information and use its additivity. That is, we obtain:

$$\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\},\omega^*)}[\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]] = \mathrm{H}''[\{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*] \tag{H.41}$$

$$= \sum_i \mathrm{H}''[{y^{\mathrm{acq}}}_i \mid {\boldsymbol{x}^{\mathrm{acq}}}_i, \omega^*]. \tag{H.42}$$

Plugging all of this together and applying Lemma 9.13, we obtain the same final approximation:

$$\mathrm{I}[\Omega; \{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}] = \dots \tag{H.43}$$

$$\approx \tfrac{1}{2}\,\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[\log\det\left(\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]\ \mathrm{H}''[\omega^*]^{-1} + Id\right)]$$

$$\leq \tfrac{1}{2}\log\det\left(\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\})}[\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]]\ \mathrm{H}''[\omega^*]^{-1} + Id\right) \tag{H.44}$$

$$\approx \tfrac{1}{2}\log\det\left(\mathbb{E}_{\mathrm{p}(\{y_i^{\mathrm{acq}}\}|\{\boldsymbol{x}_i^{\mathrm{acq}}\},\omega^*)}[\mathrm{H}''[\{y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*]]\ \mathrm{H}''[\omega^*]^{-1} + Id\right) \tag{H.45}$$

$$= \tfrac{1}{2}\log\det\left(\sum_i \mathrm{H}''[{Y^{\mathrm{acq}}}_i \mid {\boldsymbol{x}^{\mathrm{acq}}}_i, \omega^*]\ \mathrm{H}''[\omega^*]^{-1} + Id\right) \tag{H.46}$$

$$\leq \tfrac{1}{2}\sum_i \mathrm{tr}\left(\mathrm{H}''[{Y^{\mathrm{acq}}}_i \mid {\boldsymbol{x}^{\mathrm{acq}}}_i, \omega^*]\ \mathrm{H}''[\omega^*]^{-1}\right). \tag{H.47}$$

Unlike in the case of generalized linear models, a stronger assumption was necessary to reach the same result. Alternatively, we could use the GGN approximation, which leads to the same result.

H.2.2 Approximate Expected Predicted Information Gain

In the main text, we only briefly referred to not knowing a principled way to arrive at the same result of Proposition 9.16 for the general case. This is because unlike the expected information gain, the Fisher information for an acquisition candidate now lies within a matrix inversion. Even if we used the fact that $\log\det(Id + X\,Y^{-1})$ is concave in X and convex in Y, we would end up with:

$$\dots$$

$$\approx \tfrac{1}{2}\,\mathbb{E}_{\mathrm{p}(y^{\mathrm{eval}},y^{\mathrm{acq}}|\boldsymbol{x}^{\mathrm{eval}},\boldsymbol{x}^{\mathrm{acq}})\ \mathrm{p}(\boldsymbol{x}^{\mathrm{eval}})}[\log\det(\,\mathrm{H}''[y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \omega^*]\,(\mathrm{H}''[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^*])^{-1} + Id)] \tag{H.48}$$

$$\leq \tfrac{1}{2}\,\mathbb{E}_{\mathrm{p}(y^{\mathrm{acq}}|\boldsymbol{x}^{\mathrm{acq}})}[\log\det(\mathbb{E}_{\mathrm{p}(y^{\mathrm{eval}},\boldsymbol{x}^{\mathrm{eval}})}[\mathrm{H}''[y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \omega^*]]\,(\,\mathrm{H}''[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^*])^{-1} + Id)] \tag{H.49}$$

$$\geq \tfrac{1}{2}\log\det(\mathbb{E}_{\mathrm{p}(y^{\mathrm{eval}},\boldsymbol{x}^{\mathrm{eval}})}[\mathrm{H}''[y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \omega^*]]\,(\,\mathbb{E}_{\mathrm{p}(y^{\mathrm{acq}}|\boldsymbol{x}^{\mathrm{acq}})}[\mathrm{H}''[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*]] + \mathrm{H}''[\omega^*])^{-1} + Id) \tag{H.50}$$

$$= \tfrac{1}{2}\log\det(\mathbb{E}_{\mathrm{p}(\boldsymbol{x}^{\mathrm{eval}})}[\mathrm{H}''[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \omega^*]]\,(\mathrm{H}''[Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^*])^{-1} + Id) \tag{H.51}$$

$$\leq \tfrac{1}{2}\mathrm{tr}(\mathbb{E}_{\mathrm{p}(\boldsymbol{x}^{\mathrm{eval}})}[\mathrm{H}''[Y^{\mathrm{eval}} \mid \boldsymbol{x}^{\mathrm{eval}}, \omega^*]] \, (\mathrm{H}''[Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^*])^{-1}). \tag{H.52}$$

Note the $\leq \ldots \geq$, which invalidates the chain. The errors could cancel out, but a principled statement hardly seems possible using this deduction.

H.2.3 Approximate Predictive Information Gain

Similarly to Proposition 9.15, we can approximate the predictive information gain. We assume that we have access to an (empirical) distribution $\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{\mathrm{eval}}, y^{\mathrm{eval}})$:

Proposition H.3. *For a generalized linear model (or with the GGN approximation), when we take the expectation over* $\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{eval}, y^{eval})$*, we have:*

$$\arg\max_{\boldsymbol{x}^{acq}} \mathrm{I}[Y^{eval}; y^{acq} \mid X^{eval}, \boldsymbol{x}^{acq}, \mathcal{D}^{\mathrm{train}}] = \arg\min_{\boldsymbol{x}^{acq}} \mathrm{I}[\Omega; Y^{eval} \mid X^{eval}, y^{acq}, \boldsymbol{x}^{acq}, \mathcal{D}^{\mathrm{train}}] \tag{H.53}$$

with

$$\begin{aligned}
&\mathrm{I}[\Omega; Y^{eval} \mid X^{eval}, y^{acq}, \boldsymbol{x}^{acq}, \mathcal{D}^{\mathrm{train}}] && \text{(H.54)}\\
&\quad = \mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{eval}, y^{eval})} \mathrm{I}[\Omega; y^{eval} \mid \boldsymbol{x}^{eval}, y^{acq}, \boldsymbol{x}^{acq}, \mathcal{D}^{\mathrm{train}}] \\
&\quad \approx \mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{eval}, y^{eval})} [\tfrac{1}{2} \log\det(&& \text{(H.55)}\\
&\qquad \mathrm{H}''[y^{eval} \mid \boldsymbol{x}^{eval}, \omega^*] \, (\mathrm{H}''[y^{acq} \mid \boldsymbol{x}^{acq}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}])^{-1} + Id)] \\
&\quad \leq \tfrac{1}{2} \log\det(\mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{eval}, y^{eval})} [\mathrm{H}''[y^{eval} \mid \boldsymbol{x}^{eval}, \omega^*]] && \text{(H.56)}\\
&\qquad (\mathrm{H}''[y^{acq} \mid \boldsymbol{x}^{acq}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}])^{-1} + Id) \\
&\quad \leq \tfrac{1}{2} \mathrm{tr}\left(\mathbb{E}_{\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{eval}, y^{eval})} [\mathrm{H}''[y^{eval} \mid \boldsymbol{x}^{eval}, \omega^*]] \, (\mathrm{H}''[y^{acq} \mid \boldsymbol{x}^{acq}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}])^{-1} \right). && \text{(H.57)}
\end{aligned}$$

All of this follows immediately. Only for the second inequality, we need to use Jensen's inequality and that the log determinant is on the positive semi-definite symmetric cone [Cover and Thomas, 1988]. Like for the information gain, there is no difference between having access to labels or not when we have a GLM or use the GGN approximation.

H.2.4 Approximate Joint (Expected) Predictive Information Gain

A comparison of EPIG and JEPIG shows that JEPIG does not require an expectation over $\hat{\mathrm{p}}_{\mathrm{true}}(\boldsymbol{x}^{\mathrm{eval}})$ but uses a set of *evaluation samples* $\{\boldsymbol{x}_i^{\mathrm{eval}}\}$. As such, we can easily adapt Proposition 9.16 to JEPIG and obtain:

Proposition H.4 (JEPIG)**.** *For a generalized linear model (or with the GGN approximation), we have:*

$$\arg\max_{\{\boldsymbol{x}_i^{acq}\}} \mathrm{I}[\{Y_i^{eval}\}; \{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{eval}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \tag{H.58}$$

$$= \arg\min_{\{\boldsymbol{x}_i^{acq}\}} \mathrm{I}[\Omega; \{Y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \{Y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \tag{H.59}$$

with

$$\mathrm{I}[\Omega; \{Y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \{Y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}]$$

$$\approx \tfrac{1}{2}\log\det\left(\mathrm{H}''[\{Y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \omega^*]\,(\mathrm{H}''[\{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}])^{-1} + Id\right) \tag{H.60}$$

$$\leq \tfrac{1}{2}\mathrm{tr}\left(\mathrm{H}''[\{Y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \omega^*]\,(\mathrm{H}''[\{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}])^{-1}\right). \tag{H.61}$$

Similarly, for JPIG, we obtain without relying on the GGN approximation or GLMs:

Proposition H.5 (JPIG)**.** *We have:*

$$\arg\max_{\{\boldsymbol{x}_i^{acq}\}} \mathrm{I}[\{y_i^{eval}\}; \{y_i^{acq}\} \mid \{\boldsymbol{x}_i^{eval}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \tag{H.62}$$

$$= \arg\min_{\{\boldsymbol{x}_i^{acq}\}} \mathrm{I}[\Omega; \{y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \{y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}] \tag{H.63}$$

with

$$\mathrm{I}[\Omega; \{y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \{Y_i^{acq}\}, \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\text{train}}]$$

$$\approx \tfrac{1}{2}\log\det\left(\mathrm{H}''[\{y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \omega^*]\,(\mathrm{H}''[\{y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}])^{-1} + Id\right) \tag{H.64}$$

$$\leq \tfrac{1}{2}\mathrm{tr}\left(\mathrm{H}''[\{y_i^{eval}\} \mid \{\boldsymbol{x}_i^{eval}\}, \omega^*]\,(\mathrm{H}''[\{y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}])^{-1}\right). \tag{H.65}$$

Comparison between (E)PIG and J(E)PIG approximations. As observed information and Fisher information are additive, the difference between the approximations when we have an empirical, that is finite, evaluation distribution $\hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}^{\text{eval}})$ with M samples is a factor of $\mathtt{E}$ inside the log determinant or trace:

$$\mathbb{E}_{\hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}^{\text{eval}}, y^{\text{eval}})}[\mathrm{H}''[y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, \omega^*]] = \frac{1}{\mathtt{E}}\sum_i \mathrm{H}''[{y^{\text{eval}}}_i \mid {\boldsymbol{x}^{\text{eval}}}_i, \omega^*] \tag{H.66}$$

$$= \frac{1}{\mathtt{E}}\,\mathrm{H}''[\{y_i^{\text{eval}}\} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \omega^*]. \tag{H.67}$$

For the log determinant, $\frac{1}{\mathtt{E}}\log\det(A + Id) \neq \log\det(\frac{1}{\mathtt{E}}A + Id)$, but for the trace approximation, we see that both approximations are equal up to a constant factor. For example:

$$\tfrac{1}{2}\mathrm{tr}(\mathbb{E}_{\hat{\mathrm{p}}_{\text{true}}(\boldsymbol{x}^{\text{eval}})}\left[\mathrm{H}''[Y^{\text{eval}} \mid \boldsymbol{x}^{\text{eval}}, \omega^*]\right](\mathrm{H}''[\{Y_i^{\text{acq}}\} \mid \{\boldsymbol{x}_i^{\text{acq}}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}])^{-1}) \tag{H.68}$$

$$= \tfrac{1}{2}\mathrm{tr}(\frac{1}{\mathtt{E}}\mathrm{H}''[\{Y_i^{\mathrm{eval}}\} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \omega^*] \, (\mathrm{H}''[\{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}])^{-1}) \tag{H.69}$$

$$= \tfrac{1}{2\mathtt{E}}\mathrm{tr}(\mathrm{H}''[\{Y_i^{\mathrm{eval}}\} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \omega^*] \, (\mathrm{H}''[\{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}])^{-1}). \tag{H.70}$$

H.3 Similarity Matrices and One-Sample Approximations

Proposition 9.17. *Given $\mathcal{D}^{\mathrm{train}}$, $\{\boldsymbol{x}_i^{acq}\}$ and (sampled) $\{y_i^{acq}\}$, we have for the EIG:*

$$\mathrm{I}[\Omega; \{Y_i^{acq}\} \mid \{\boldsymbol{x}_i^{acq}\}, \mathcal{D}^{\mathrm{train}}] \lessapprox \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]}[\mathcal{D}^{\mathrm{acq}} \mid \omega^*] + Id\right) \tag{9.66}$$

$$\leq \tfrac{1}{2}\mathrm{tr}\, S_{\mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]}[\mathcal{D}^{\mathrm{acq}} \mid \omega^*] \tag{9.67}$$

Proof.

$$\mathrm{I}[\Omega; \{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \mathcal{D}^{\mathrm{train}}] \lessapprox \tfrac{1}{2}\log\det\left(\mathrm{H}''[\{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*] \; \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]^{-1} + Id\right) \tag{H.71}$$

$$= \tfrac{1}{2}\log\det\left((\mathrm{H}''[\{Y_i^{\mathrm{acq}}\} \mid \{\boldsymbol{x}_i^{\mathrm{acq}}\}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]) \; \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]^{-1}\right) \tag{H.72}$$

$$\approx \tfrac{1}{2}\log\det\left((\hat{\mathrm{H}}'[\mathcal{D}^{\mathrm{acq}} \mid \omega^*]^T \hat{\mathrm{H}}'[\mathcal{D}^{\mathrm{acq}} \mid \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]) \; \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]^{-1}\right) \tag{H.73}$$

$$= \tfrac{1}{2}\log\det\left(\hat{\mathrm{H}}'[\mathcal{D}^{\mathrm{acq}} \mid \omega^*] \, \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]^{-1} \hat{\mathrm{H}}'[\mathcal{D}^{\mathrm{acq}} \mid \omega^*]^T + Id\right) \tag{H.74}$$

$$= \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]}[\mathcal{D}^{\mathrm{acq}} \mid \omega^*] + Id\right), \tag{H.75}$$

where we have used the matrix determinant lemma:

$$\det(AB + M) = \det(BM^{-1}A + Id) \, \det M. \tag{H.76}$$

□

Connection to the Joint (Expected) Predictive Information Gain. Following Equation 9.49, JEPIG can be decomposed as the difference between two EIG terms, which we can further divide into three terms that are only conditioned on $\mathcal{D}^{\mathrm{train}}$:

$$\mathrm{I}[\{Y_i^{\mathrm{eval}}\}; Y^{\mathrm{acq}} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \boldsymbol{x}^{\mathrm{acq}}, \mathcal{D}^{\mathrm{train}}] \tag{H.77}$$

$$= \mathrm{I}[\Omega; \{Y_i^{\mathrm{eval}}\} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \mathcal{D}^{\mathrm{train}}] - \mathrm{I}[\Omega; \{Y_i^{\mathrm{eval}}\} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, Y^{\mathrm{acq}}, \boldsymbol{x}^{\mathrm{acq}}, \mathcal{D}^{\mathrm{train}}]$$

$$= \mathrm{I}[\Omega; \{Y_i^{\mathrm{eval}}\} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \mathcal{D}^{\mathrm{train}}] - \mathrm{I}[\Omega; \{Y_i^{\mathrm{eval}}\}, Y^{\mathrm{acq}} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \boldsymbol{x}^{\mathrm{acq}}, \mathcal{D}^{\mathrm{train}}] \tag{H.78}$$

$$+ \mathrm{I}[\Omega; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \mathcal{D}^{\mathrm{train}}]$$

Using Proposition 9.17, we can approximate this as:

$$\mathrm{I}[\{Y_i^{\mathrm{eval}}\}; Y^{\mathrm{acq}} \mid \{\boldsymbol{x}_i^{\mathrm{eval}}\}, \boldsymbol{x}^{\mathrm{acq}}, \mathcal{D}^{\mathrm{train}}] \tag{H.79}$$

$$\approx \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]}[\mathcal{D}^{\mathrm{eval}} \mid \omega^*] + Id\right) - \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]}[\mathcal{D}^{\mathrm{acq}}, \mathcal{D}^{\mathrm{eval}} \mid \omega^*] + Id\right) \tag{H.80}$$

$$+ \tfrac{1}{2}\log\det\left(S_{\mathrm{H}''[\omega^*|\mathcal{D}^{\text{train}}]}[\mathcal{D}^{\text{acq}} \mid \omega^*] + Id\right) \tag{H.81}$$

Furthermore, we can apply the approximation in Proposition 9.18 and find that the $\log\lambda$ terms cancel because $|\mathcal{D}^{\text{acq}}| + |\mathcal{D}^{\text{train}}| = |\mathcal{D}^{\text{acq}} \cup \mathcal{D}^{\text{train}}|$. Taking the limit $\lambda \to 0$, we obtain:

$$\mathrm{I}[\{Y_i^{\text{eval}}\}; Y^{\text{acq}} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] \tag{H.82}$$

$$\approx \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{eval}} \mid \omega^*] + \lambda Id\right) - \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{acq}}, \mathcal{D}^{\text{eval}} \mid \omega^*] + \lambda Id\right) \tag{H.83}$$

$$+ \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{acq}} \mid \omega^*] + \lambda Id\right)$$

$$\to \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{eval}} \mid \omega^*]\right) - \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{acq}}, \mathcal{D}^{\text{eval}} \mid \omega^*]\right) + \tfrac{1}{2}\log\det\left(S[\mathcal{D}^{\text{acq}} \mid \omega^*]\right). \tag{H.84}$$

Finally, the first term is independent of $\mathcal{D}^{\text{acq}}$, and if we are interested in approximately maximizing JEPIG, we can maximize as proxy objective:

$$\log\det\left(S_{\mathrm{H}''[\omega^*|\mathcal{D}^{\text{train}}]}[\mathcal{D}^{\text{acq}} \mid \omega^*] + Id\right) - \log\det\left(S_{\mathrm{H}''[\omega^*|\mathcal{D}^{\text{train}}]}[\mathcal{D}^{\text{acq}}, \mathcal{D}^{\text{eval}} \mid \omega^*] + Id\right), \tag{H.85}$$

or

$$\log\det\left(S[\mathcal{D}^{\text{acq}} \mid \omega^*]\right) - \log\det\left(S[\mathcal{D}^{\text{acq}}, \mathcal{D}^{\text{eval}} \mid \omega^*]\right). \tag{H.86}$$

H.4 Connection to Other Acquisition Functions in the Literature

H.4.1 SIMILAR [Kothawade et al., 2021] and PRISM [Kothawade et al., 2022]

Connection to LogDetMI. If we apply the Schur decomposition to $\log\det S[\mathcal{D}^{\text{acq}}, \mathcal{D}^{\text{eval}} \mid \omega^*]$ from Equation H.86, we obtain the following:

$$\log\det S[\mathcal{D}^{\text{acq}} \mid \omega^*] - \log\det S[\mathcal{D}^{\text{acq}}, \mathcal{D}^{\text{eval}} \mid \omega^*] \tag{H.87}$$

$$= \log\det S[\mathcal{D}^{\text{acq}} \mid \omega^*] - \log\det S[\mathcal{D}^{\text{eval}} \mid \omega^*] \tag{H.88}$$

$$- \log\det(S[\mathcal{D}^{\text{acq}} \mid \omega^*] - S[\mathcal{D}^{\text{acq}}; \mathcal{D}^{\text{eval}} \mid \omega^*] S[\mathcal{D}^{\text{eval}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{eval}}; \mathcal{D}^{\text{acq}} \mid \omega^*]),$$

where $S[\mathcal{D}^{\text{acq}}; \mathcal{D}^{\text{eval}} \mid \omega^*]$ is the non-symmetric similarity matrix between $\mathcal{D}^{\text{acq}}$ and $\mathcal{D}^{\text{eval}}$ etc.

Dropping $\log\det S[\mathcal{D}^{\text{eval}} \mid \omega^*]$ which is independent of $\mathcal{D}^{\text{acq}}$, we can instead maximize:

$$\log\det S[\mathcal{D}^{\text{acq}} \mid \omega^*] - \log\det(S[\mathcal{D}^{\text{acq}} \mid \omega^*] \tag{H.89}$$

$$- S[\mathcal{D}^{\text{acq}}; \mathcal{D}^{\text{eval}} \mid \omega^*] S[\mathcal{D}^{\text{eval}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{eval}}; \mathcal{D}^{\text{acq}} \mid \omega^*],)$$

which is exactly the LogDetMI objective of SIMILAR [Kothawade et al., 2021] and PRISM [Kothawade et al., 2022].

We can further rewrite this objective by extracting $S[\mathcal{D}^{\text{acq}} \mid \omega^*]$ from the second term, obtaining:

$$\log\det S[\mathcal{D}^{\text{acq}} \mid \omega^*] - \log\det(S[\mathcal{D}^{\text{acq}} \mid \omega^*]$$

$$- S[\mathcal{D}^{\text{acq}}; \mathcal{D}^{\text{eval}} \mid \omega^*] S[\mathcal{D}^{\text{eval}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{eval}}; \mathcal{D}^{\text{acq}} \mid \omega^*]) \tag{H.90}$$

$$= -\log\det(Id - S[\mathcal{D}^{\text{acq}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{acq}}; \mathcal{D}^{\text{eval}} \mid \omega^*] S[\mathcal{D}^{\text{eval}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{eval}}; \mathcal{D}^{\text{acq}} \mid \omega^*]). \tag{H.91}$$

Connection to LogDetCMI. Using information-theoretic decompositions, it is easy to show that:

$$\mathrm{I}[\{Y_i^{\text{eval}}\}; Y^{\text{acq}} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \boldsymbol{x}^{\text{acq}}, \{Y_i\}, \{\boldsymbol{x}_i\}, \mathcal{D}^{\text{train}}] \tag{H.92}$$

$$= \mathrm{I}[\{Y_i^{\text{eval}}\}; Y^{\text{acq}}, \{Y_i\} \mid \{\boldsymbol{x}_i^{\text{eval}}\}, \boldsymbol{x}^{\text{acq}}, \{\boldsymbol{x}_i\}, \mathcal{D}^{\text{train}}] - \mathrm{I}[\{Y_i^{\text{eval}}\}; \{Y_i\} \mid \{\boldsymbol{x}_i\}, \mathcal{D}^{\text{train}}]. \tag{H.93}$$

These are two JEPIG terms, and using above approximations, including (H.91), leads to the LogDetCMI objective:

$$\log \frac{\det(Id - S[\mathcal{D}^{\text{acq}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{acq}}; \mathcal{D}^{\text{eval}} \mid \omega^*] S[\mathcal{D}^{\text{eval}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{eval}}; \mathcal{D}^{\text{acq}} \mid \omega^*])}{\det(Id - S[\mathcal{D}^{\text{acq}}, \mathcal{D} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{acq}}, \mathcal{D}; \mathcal{D}^{\text{eval}} \mid \omega^*] S[\mathcal{D}^{\text{eval}} \mid \omega^*]^{-1} S[\mathcal{D}^{\text{eval}}; \mathcal{D}^{\text{acq}}, \mathcal{D} \mid \omega^*])}. \tag{H.94}$$

H.4.2 Expected Gradient Length

Proposition 9.23. *The EIG for a candidate sample $\boldsymbol{x}^{acq}$ approximately lower-bounds the EGL:*

$$2\,\mathrm{I}[\Omega; Y^{acq} \mid \boldsymbol{x}^{acq}] \overset{\approx}{\leq} \mathbb{E}_{\mathrm{p}(y^{acq} \mid \boldsymbol{x}^{acq}, \omega^*)} \left\| \mathrm{H}'[y^{acq} \mid \boldsymbol{x}^{acq}, \omega^*] \right\|^2 + const. \tag{9.78}$$

Proof. The EIG is equal to the conditional entropy up to a constant term, via Equation 9.43 in Proposition 9.14:

$$\mathrm{I}[\Omega; Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}] \overset{\approx}{\leq} \tfrac{1}{2} \log\det\left(\mathrm{H}''[Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]\right) + \text{const.} \tag{H.95}$$

We apply a diagonal approximation for the Fisher information and Hessian, noting that the determinant of the diagonal matrix upper-bounds the determinant of the full matrix:

$$\leq \tfrac{1}{2} \log\det\left(\mathrm{H}''_{diag}[Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*] + \mathrm{H}''_{diag}[\omega^* \mid \mathcal{D}^{\text{train}}]\right) + \text{const.} \tag{H.96}$$

$$= \tfrac{1}{2} \sum_k \log\left(\mathrm{H}''_{diag,kk}[Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*] + \mathrm{H}''_{diag,kk}[\omega^* \mid \mathcal{D}^{\text{train}}]\right) + \text{const.} \tag{H.97}$$

We use $\log x \leq x - 1$ and that $\mathrm{H}''[\omega^* \mid \mathcal{D}^{\text{train}}]$ is constant:

$$\leq \tfrac{1}{2} \sum_k \left(\mathrm{H}''_{diag,kk}[Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*] + \mathrm{H}''_{diag,kk}[\omega^* \mid \mathcal{D}^{\text{train}}]\right) + \text{const.} \tag{H.98}$$

$$\leq \tfrac{1}{2} \sum_k \mathrm{H}''_{diag,kk}[Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*] + \text{const.} \tag{H.99}$$

From Proposition 9.7, we know that the Fisher information is equivalent to the outer product of the Jacobians: $\mathrm{H}''[Y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*] = \mathbb{E}_{\mathrm{p}(y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*)}[\mathrm{H}'[y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*]\ \mathrm{H}'[y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*]^T]$, and we finally obtain for the diagonal elements:

$$= \tfrac{1}{2} \sum_k \mathbb{E}_{\mathrm{p}(y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*)} \left[\mathrm{H}'_k[y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*]^2\right] + \text{const.} \tag{H.100}$$

$$= \tfrac{1}{2}\, \mathbb{E}_{\mathrm{p}(y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*)} \left[\left\| \mathrm{H}'[y^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \omega^*] \right\|^2\right] + \text{const.} \tag{H.101}$$

□

H.4.3 Deep Learning on a Data Diet

Proposition 9.24. *The IG for a candidate sample $\boldsymbol{x}^{acq}$ approximately lower-bounds the gradient norm score (GraNd) at ω^* up to a second-order term:*

$$2\,\mathrm{I}[\Omega; y^{acq} \mid \boldsymbol{x}^{acq}] \overset{\approx}{\leq} \mathbb{E}_{\mathrm{q}(\omega)}[\left\|\mathrm{H}'[y^{acq} \mid \boldsymbol{x}^{acq}, \omega]\right\|^2] - \mathbb{E}_{\mathrm{q}(\omega)}[\mathrm{tr}\left(\frac{\nabla^2_\omega\, \mathrm{p}(y \mid x, \omega)}{\mathrm{p}(y \mid x, \omega)}\right)] + \textit{const.} \tag{9.79}$$

Proof. For any fixed ω^*, the IG is equal to the conditional entropy up to a constant term, via Proposition 9.15:

$$\mathrm{I}[\Omega; Y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}] \overset{\approx}{\leq} \tfrac{1}{2} \log\det\left(\mathrm{H}''[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]\right) + \mathrm{const.} \tag{H.102}$$

As in the previous proof, we apply a diagonal approximation for the Hessian, noting that the determinant of the diagonal matrix upper-bounds the determinant of the full matrix:

$$\leq \tfrac{1}{2} \log\det\left(\mathrm{H}''_{diag}[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''_{diag}[\omega^* \mid \mathcal{D}^{\mathrm{train}}]\right) + \mathrm{const.} \tag{H.103}$$

$$= \tfrac{1}{2} \sum_k \log\left(\mathrm{H}''_{diag,kk}[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''_{diag,kk}[\omega^* \mid \mathcal{D}^{\mathrm{train}}]\right) + \mathrm{const.} \tag{H.104}$$

Again, we use $\log x \leq x - 1$ and that $\mathrm{H}''[\omega^* \mid \mathcal{D}^{\mathrm{train}}]$ is constant:

$$\leq \tfrac{1}{2} \sum_k \left(\mathrm{H}''_{diag,kk}[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{H}''_{diag,kk}[\omega^* \mid \mathcal{D}^{\mathrm{train}}]\right) + \mathrm{const.} \tag{H.105}$$

$$\leq \tfrac{1}{2} \sum_k \mathrm{H}''_{diag,kk}[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] + \mathrm{const.} \tag{H.106}$$

From Lemma H.1, we know that the Hessian is equivalent to the outer product of the Jacobians plus a second-order term: $\mathrm{H}''[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*] = \mathrm{H}'[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*]\ \mathrm{H}'[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*]^T - \frac{\nabla^2_\omega\, \mathrm{p}(y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*)}{\mathrm{p}(y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*)}$, and we finally obtain for the diagonal elements:

$$= \tfrac{1}{2} \sum_k \mathrm{H}'_k[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*]^2 - \tfrac{1}{2}\,\mathrm{tr}\left(\frac{\nabla^2_\omega\, \mathrm{p}(y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*)}{\mathrm{p}(y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*)}\right) + \mathrm{const.} \tag{H.107}$$

$$= \tfrac{1}{2}\left\|\mathrm{H}'[y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*]\right\|^2 - \tfrac{1}{2}\,\mathrm{tr}\left(\frac{\nabla^2_\omega\, \mathrm{p}(y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*)}{\mathrm{p}(y^{\mathrm{acq}} \mid \boldsymbol{x}^{\mathrm{acq}}, \omega^*)}\right) + \mathrm{const.} \tag{H.108}$$

Taking an expectation over $\omega^* \sim \mathrm{q}(\omega)$ yields the statement. □

H.5 Preliminary Empirical Comparison of Information Quantity Approximations

The following section describes an initial empirical evaluation[1] of the bounds from §9.4 on MNIST. We train a model on a subset of MNIST and compare the approximations

[1] Code at: https://github.com/BlackHC/2208.00549

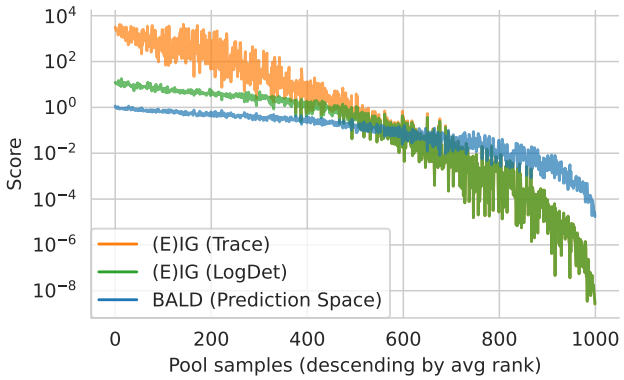


Figure H.1: *EIG Approximations.* Trace and log det approximations match for small scores (because of Lemma 9.13). They diverge for large scores. Qualitatively, the order matches the prediction-space approximation using BALD with MC dropout.

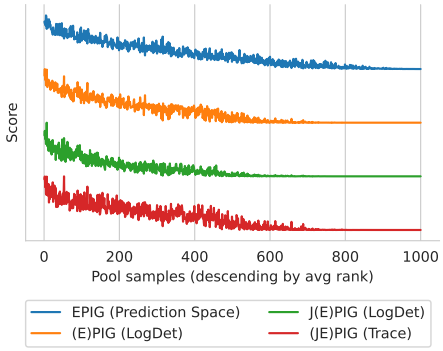


Figure H.2: *(J)EPIG Approximations (Normalized).* The scores match qualitatively. Note we have reversed the ordering for the proxy objectives for JEPIG and EPIG as they are minimized while EPIG is maximized.

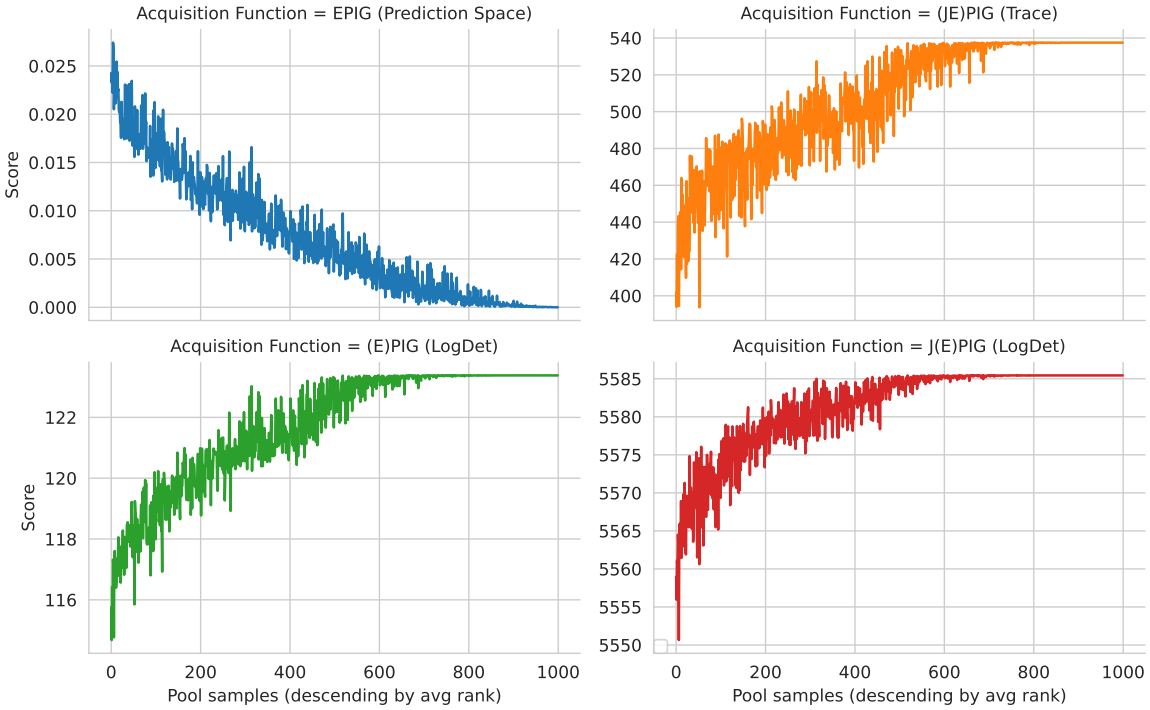


Figure H.3: *(J)EPIG Approximations.* The scores match quantitatively. Note the proxy objectives for JEPIG and EPIG are minimized while EPIG is maximized. The value ranges are off by a lot: the true EPIG score is upper bounded by $\log \mathtt{C} \approx 2.3\,nats$.

Table H.1: *Spearman Rank Correlation of Prediction-Space and Weight-Space Estimates.* BALD and EPIG are both strongly positively rank-correlated with each other. The weight-space approximations are strongly rank-correlated with the prediction-space approximations, but the weight-space approximations are less accurate than the prediction-space approximations. Note that we have reversed the ordering for the proxy objectives for JEPIG and EPIG as they are minimized while EPIG is maximized.

	BALD (Prediction)	EIG (LogDet)	EIG (Trace)	EPIG (Prediction)	EPIG (LogDet)	JEPIG (LogDet)	(J)EPIG (Trace)
BALD (Prediction)	1.000	0.955	0.940	0.984	0.948	0.955	0.927
EPIG (Prediction)	0.984	0.918	0.897	1.000	0.918	0.918	0.903

of EIG and EPIG in weight space to BALD and EPIG computed in prediction space. We use a last-layer approach (GLM) which means that active sampling and active learning approximations are equivalent. We do not attempt to estimate JEPIG in prediction space.

Setup. We train a BNN using MC dropout on 80 randomly selected training samples from MNIST, achieving 83% accuracy. The model architecture follows the one described in §4. We use 100 Monte-Carlo dropout samples [Gal and Ghahramani, 2016a] to compute the prediction-space estimates. For EPIG, we sample the evaluation set from the remaining training set (20000 samples). We randomly select 1000 samples from the training set as pool set. We compute BALD and EPIG in prediction space as described in Gal et al. [2017] and §7. For the weight-space approximations, we use a last-layer approximation—we thus have a GLM. For the implementation, we use PyTorch [Paszke et al., 2019] and the laplace-torch library [Daxberger et al., 2021].

We chose 80 samples and 83% accuracy as the accuracy trajectory of BALD and EPIG is steep at this point, see e.g. §4, and thus we expect a wider range of scores.

Results. In Figure H.1, we see a comparison of BALD with the approximations in Equation 9.40 and Equation 9.41. Not shown is Equation 9.43, which performs like Equation 9.40 (up to a constant). In Figure H.2 and Figure H.3, we show a comparison of EPIG with the approximations in Equation 9.58, Equation 9.59, and Equation H.60. Figure H.2 shows normalized scores individually as the score ranges are very different.

Importantly, while the prediction-space scores (BALD and EPIG) have valid scores as the EIG/BALD and EPIG scores of a sample are bounded by the $\log \mathtt{C}$, the weight-space scores are not valid. As such, they only provide rough estimates of the information quantities.

However, as we see in Table H.1, the Spearman rank correlation coefficients between the weight-space and prediction-space scores are very high. Thus, while the scores themselves are not good estimates, their order seems informative, and this is what matters for selecting acquisition samples in data subset selection.

Future Work. The quality of these approximations needs further verification using more complex models and datasets. Comparisons in active learning and active sampling experiments are also necessary to validate the usefulness of these approximations. However, given the connections shown in §9.6, we expect that the approximations will be useful in these settings as well.

I

Black-Box Batch Active Learning for Regression

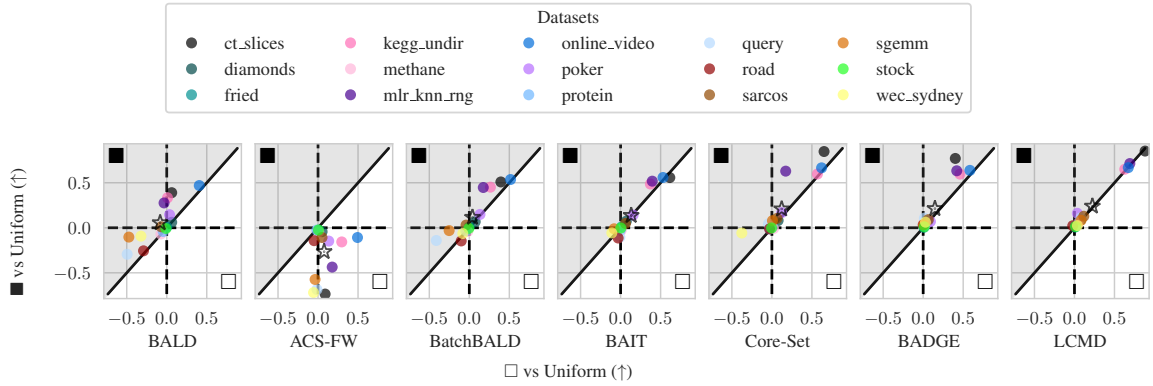
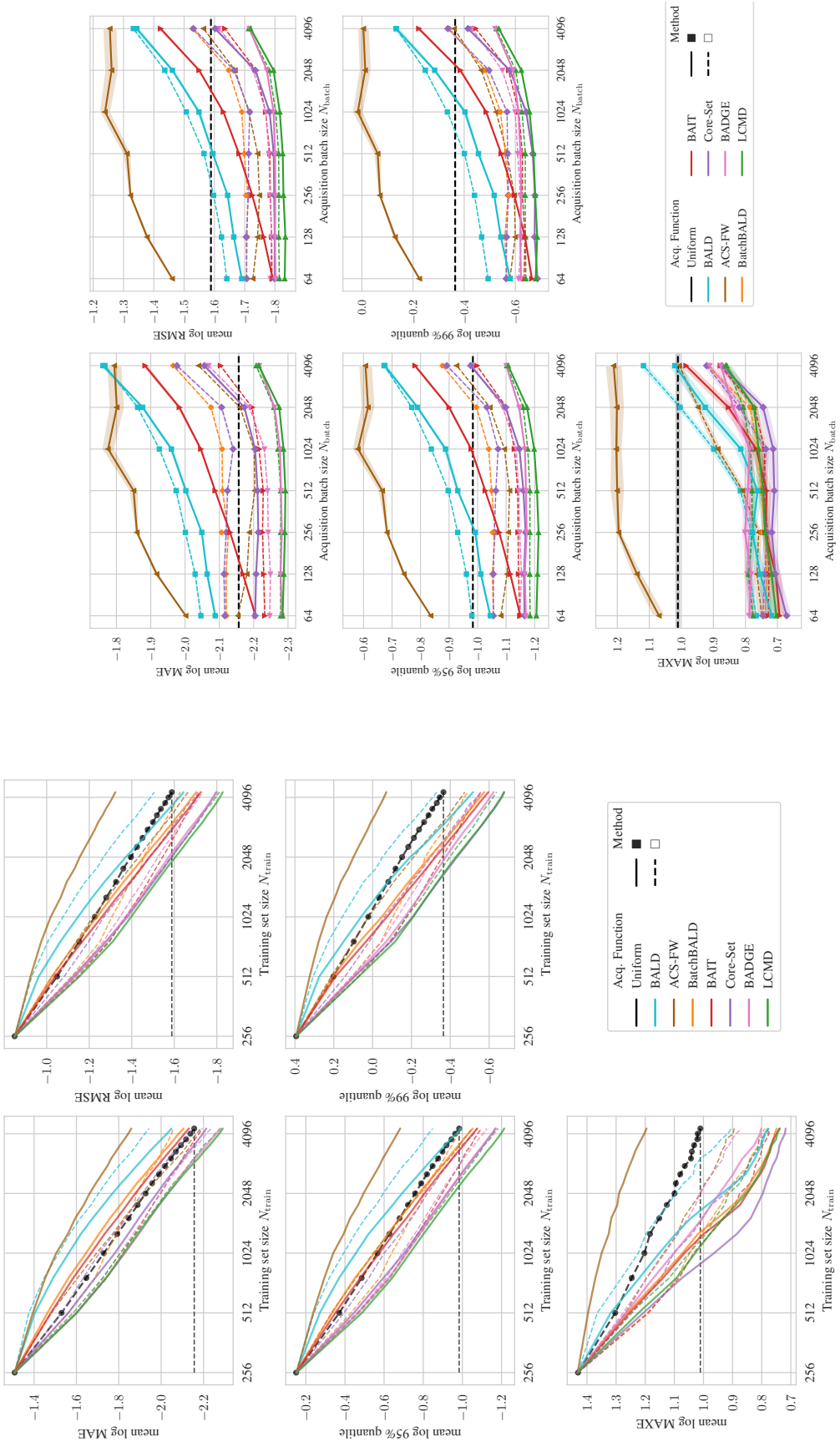


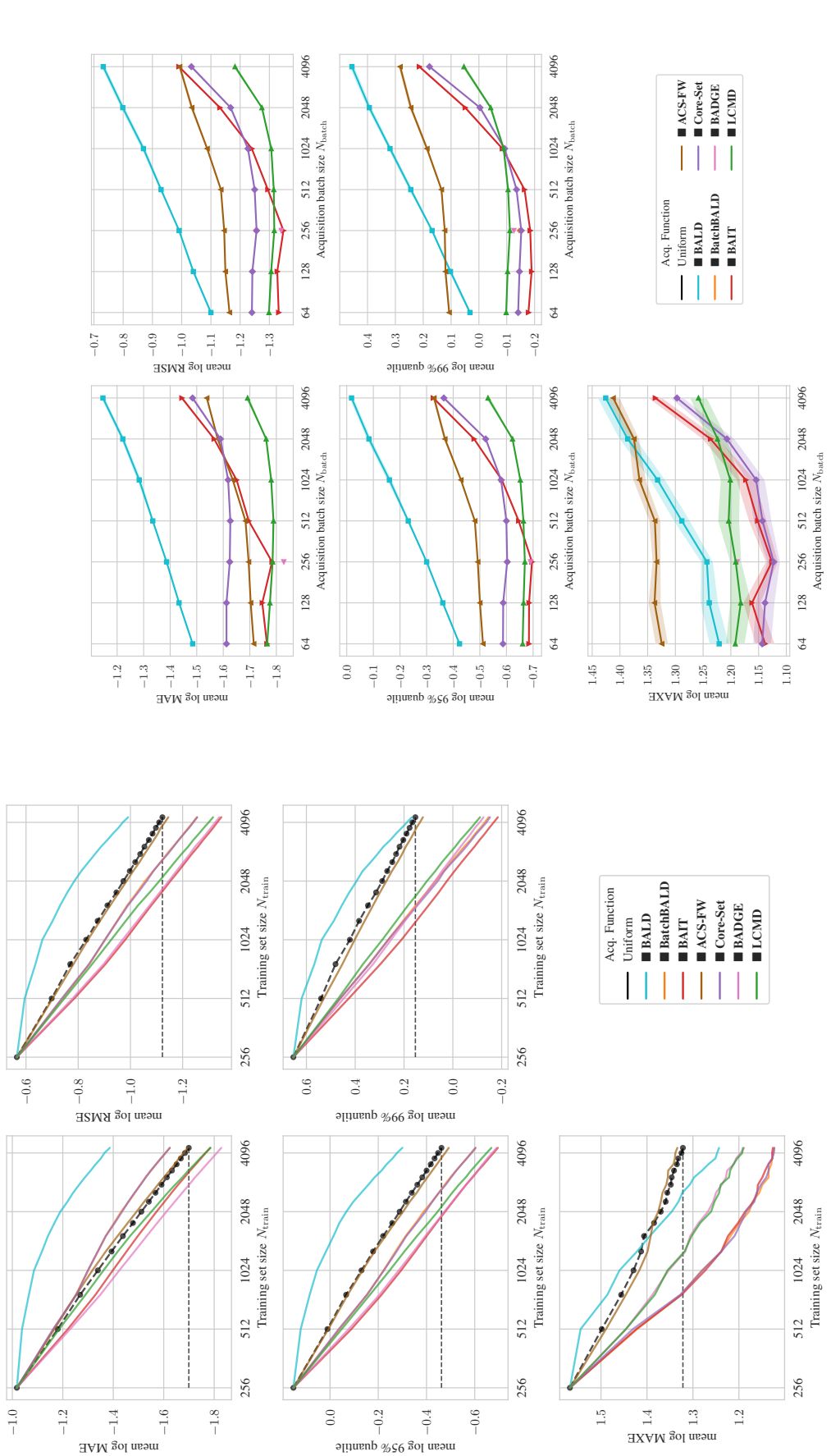
Figure I.1: *Final Logarithmic RMSE by regression datasets for DNNs: ■ vs □ (vs Uniform).* Across acquisition functions, the performance of black-box methods is highly correlated with the performance of white-box methods, even though black-box methods make fewer assumptions about the model. We plot the improvement of the white-box method (□) over the uniform baseline on the x-axis, so for datasets with markers left of the dashed vertical lines, the white-box method performs better than uniform, and the improvement of the black-box method (■) over the uniform baseline on the y-axis, so for datasets with markers above the dashed horizontal lines, the black-box method performs better than uniform. Similarly, for datasets with markers in the ■ region, the black-box method performs better than the white-box method. The average over all datasets is marked with a star ⋆.



(a) Learning Curves

(b) Acquisition Batch Sizes

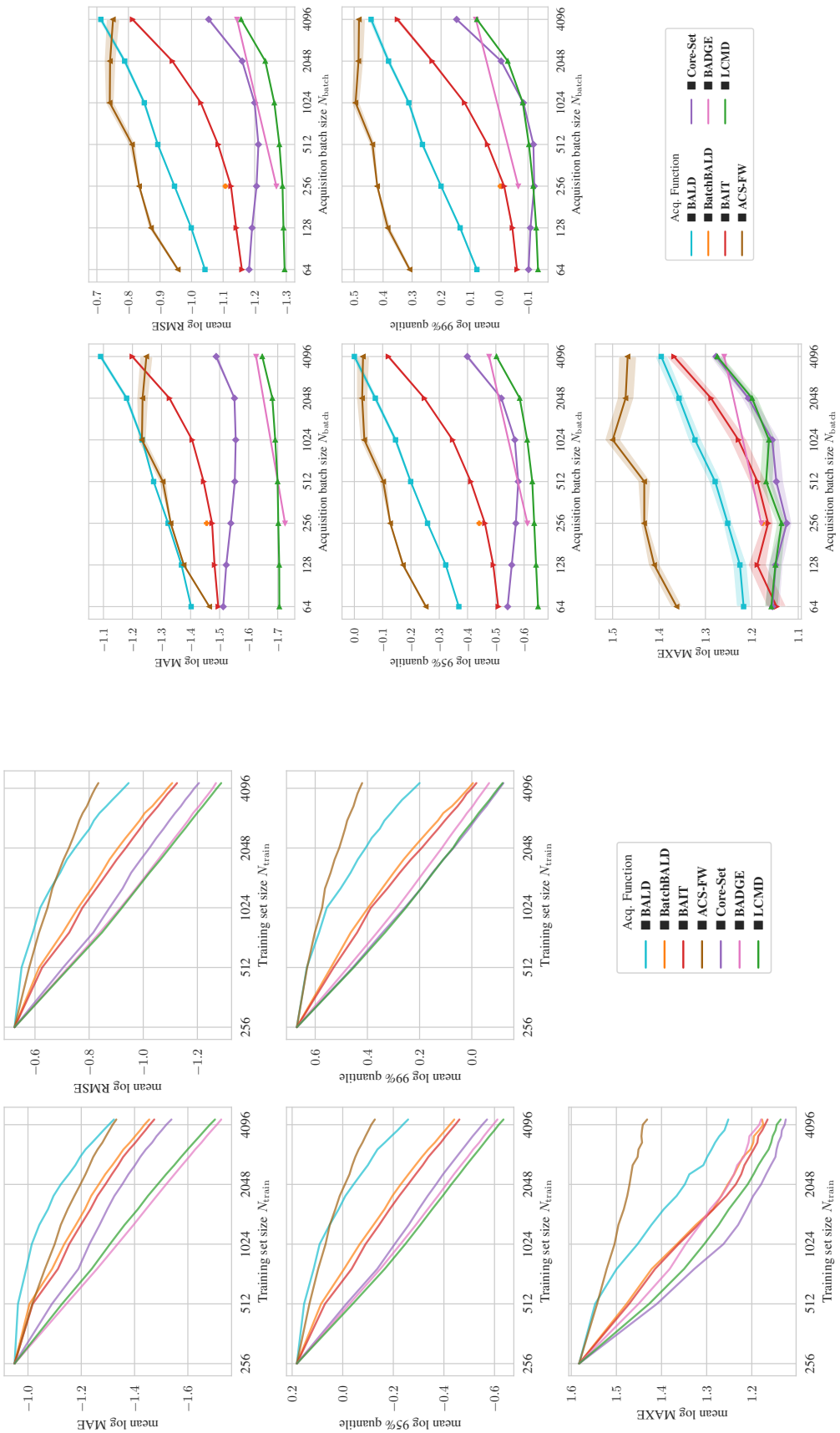
Figure I.2: *DNNs: Error Metrics over 15 regression datasets.* We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 20 trials.



(a) Learning Curves

(b) Acquisition Batch Size

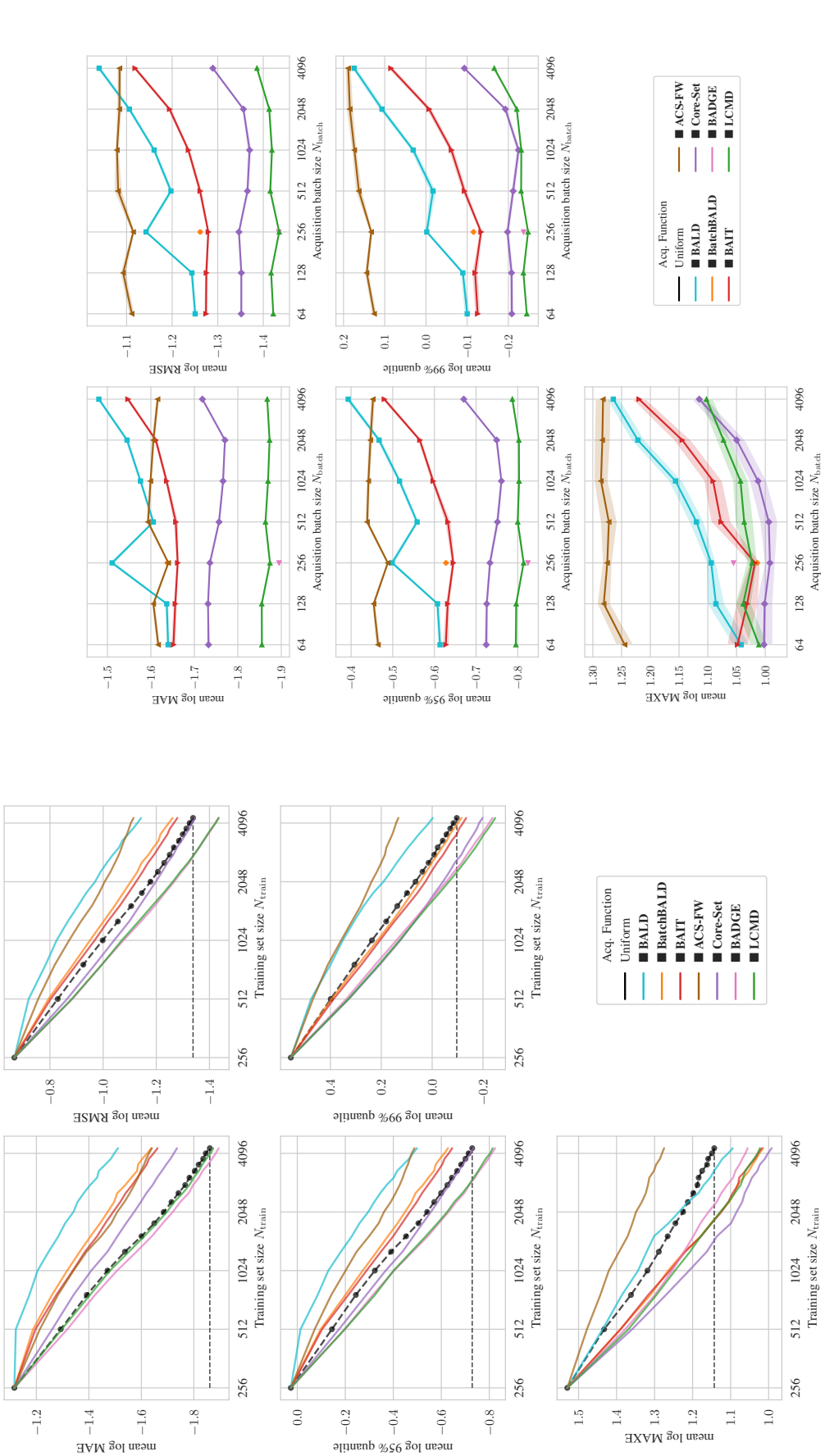
Figure I.3: *Random Forests: Error Metrics over 15 regression datasets (cont'd).* We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 20 trials.



(a) Learning Curves

(b) Acquisition Batch Size

Figure I.4: *Random Forests (Bagging): Error Metrics over 15 regression datasets (cont'd).* We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 20 trials.



(a) Learning Curves **(b)** Acquisition Batch Size

Figure I.5: *Gradient-Boosted Trees with Virtual Ensemble: Error Metrics over 15 regression datasets (cont'd).* We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 20 trials.

Table I.1: *Average performance of black-box* ■ *and white-box* □ *batch active learning acquisition functions using DNNs.* On average, for five acquisition methods, the black-box method performs better than the white-box method. Cf. Figure 10.3, which analyzes the final epoch.

Acquisition function	MAE	RMSE	95%	99%	MAXE
Uniform	-1.934	-1.401	-0.766	-0.163	1.107
■ **BALD**	-1.794	-1.389	-0.713	-0.221	0.946
□ BALD	-1.722	-1.285	-0.614	-0.077	1.080
■ **BatchBALD**	-1.865	-1.465	-0.792	-0.303	0.892
□ BatchBALD	-1.895	-1.463	-0.808	-0.288	0.916
□ BAIT	-1.998	-1.541	-0.895	-0.357	0.888
■ **BAIT**	-1.892	-1.489	-0.817	-0.328	0.881
□ ACS-FW	-1.937	-1.439	-0.793	-0.225	1.016
■ **ACS-FW**	-1.678	-1.168	-0.509	0.085	1.278
■ **Core-Set**	-1.988	-1.585	-0.926	-0.435	**0.831**
□ Core-Set	-1.923	-1.490	-0.831	-0.307	0.929
■ **BADGE**	-2.042	-1.579	-0.931	-0.383	0.948
□ BADGE	-2.007	-1.530	-0.895	-0.329	1.008
■ **LCMD**	**-2.048**	**-1.609**	**-0.965**	**-0.437**	0.874
□ LCMD	-2.033	-1.589	-0.940	-0.402	0.914

Table I.2: *Average performance of black-box* ■ *batch active learning acquisition functions on non-differentiable models.*

(a) Random Forests

Acquisition function	MAE	RMSE	95%	99%	MAXE
Uniform	-1.516	-0.975	-0.293	0.290	1.376
■ **BALD**	-1.222	-0.815	-0.106	0.365	1.348
■ **BatchBALD**	-1.449	-1.070	-0.401	0.064	**1.195**
■ **BAIT**	-1.582	**-1.158**	**-0.485**	**0.021**	1.198
■ **ACS-FW**	-1.502	-0.989	-0.310	0.266	1.379
■ **Core-Set**	-1.449	-1.071	-0.403	0.059	1.196
■ **BADGE**	**-1.618**	-1.150	-0.480	0.070	1.273
■ **LCMD**	-1.564	-1.116	-0.450	0.097	1.270

(b) Gradient-Boosted Decision Trees

Acquisition function	MAE	RMSE	95%	99%	MAXE
Uniform	-1.675	-1.172	-0.533	0.068	1.232
■ **BALD**	-1.353	-0.979	-0.308	0.182	1.225
■ **BatchBALD**	-1.474	-1.103	-0.448	0.052	1.134
■ **BAIT**	-1.497	-1.122	-0.467	0.034	1.137
■ **ACS-FW**	-1.501	-1.000	-0.354	0.241	1.346
■ **Core-Set**	-1.573	-1.193	-0.552	-0.036	**1.100**
■ **BADGE**	**-1.709**	**-1.259**	**-0.621**	-0.050	1.156
■ **LCMD**	-1.688	-1.256	-0.616	**-0.061**	1.132

Table I.3: *Overview over the used datasets.* See Ballester-Ripoll et al. [2019]; Neshat et al. [2018]; Graf et al. [2011]; Shannon et al. [2003]; Deneke et al. [2014]; Anagnostopoulos et al. [2018]; Savva et al. [2018]; Friedman [1991]; Ślęzak et al. [2018]. The second column entries are hyperlinks to the respective web pages. Taken from Holzmüller et al. [2022].

Short name	Initial pool set size	Test set size	Number of features	Source	OpenML ID	Full name
sgemm	192000	48320	14	UCI		SGEMM GPU kernel performance
wec_sydney	56320	14400	48	UCI		Wave Energy Converters
ct_slices	41520	10700	379	UCI		Relative location of CT slices on axial axis
kegg_undir	50407	12921	27	UCI		KEGG Metabolic Reaction Network (Undirected)
online_video	53748	13756	26	UCI		Online Video Characteristics and Transcoding Time
query	158720	40000	4	UCI		Query Analytics Workloads
poker	198720	300000	95	UCI		Poker Hand
road	198720	234874	2	UCI		3D Road Network (North Jutland, Denmark)
mlr_knn_rng	88123	22350	132	OpenML	42454	mlr_knn_rng
fried	31335	8153	10	OpenML	564	fried
diamonds	41872	10788	29	OpenML	42225	diamonds
methane	198720	300000	33	OpenML	42701	Methane
stock	45960	11809	9	OpenML	1200	BNG(stock)
protein	35304	9146	9	OpenML	42903	physicochemical-protein
sarcos	34308	8896	21	GPML		SARCOS data

J Contributions to Joint Work

The following declaration of contributions summarizes my contributions to the papers that are used as a basis for my thesis. Several of the papers are joint-authored with me as joint first-author. Prof. Yarin Gal supervised all my research projects except for Kirsch [2023b] and Kirsch [2023a].

A Practical & Unified Notation for Information Quantities with Observed Outcomes [Kirsch and Gal, 2021]

> Andreas Kirsch and Yarin Gal. A Practical & Unified Notation for Information-Theoretic Quantities in ML. *arXiv preprint*, 2021

Contributions. I developed the paper and idea.

Deep Deterministic Uncertainty: A Simple Baseline [Mukhoti et al., 2023]

> Jishnu Mukhoti*, Andreas Kirsch*, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023

Contributions. Jishnu and I co-authored the paper. Jishnu and I wrote the paper together. I contributed significantly to the theory, in particular the observations, and suggested using density scores for OoD detection (using GDA) instead of scores based on the predictive distribution (entropy). Yarin and I developed the idea of the Dirty-MNIST dataset. Jishnu implemented and ran all experiments and trained all the models while Joost and I provided code reviews and feedback.

BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning [Kirsch et al., 2019]

> Andreas Kirsch*, Joost van Amersfoort*, and Yarin Gal. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *Advances in Neural Information Processing Systems*, 2019

Contributions. Joost and I are joint first-authors. I developed the idea and algorithm and implemented and ran the experiments. Joost developed the Repeated-MNIST experimental setting and came up with the acquisition size-time plot. Joost and I co-wrote the paper. I wrote the proofs and created the plots.

Stochastic Batch Acquisition: A Simple Baseline for Deep Active Learning [Kirsch et al., 2023]

> Andreas Kirsch*, Sebastian Farquhar*, Parmida Atighehchian, Andrew Jesson, Frederic Branchaud-Charron, and Yarin Gal. Stochastic Batch Acquisition for Deep Active Learning. *Transactions on Machine Learning Research*, 2023

Contributions. Sebastian and I are joint first-authors. I implemented and ran the initial experiments and wrote the workshop submission while Sebastian was on an internship. The initial idea of stochastic acquisition was independently and then jointly developed. Sebastian designed the rank correlation experiment, and we jointly rewrote the paper with him leading and editing the paper. I created the plots and ran experiments. Frederic, Parmida, and Andrew ran experiments.

Marginal and Joint Cross-Entropies & Predictives for Online Bayesian Inference, Active Learning, and Active Sampling [Kirsch et al., 2022]

> Andreas Kirsch, Jannik Kossen, and Yarin Gal. Marginal and Joint Cross-Entropies & Predictives for Online Bayesian Inference, Active Learning, and Active Sampling. *arXiv preprint*, 2022

Contributions. I was the lead author on the workshop submission and the preprint. Jannik supported the project's experiment design and provided feedback and helped edit the paper.

Test Distribution–Aware Active Learning: A Principled Approach Against Distribution Shift and Outliers [Kirsch et al., 2021]

> Andreas Kirsch, Tom Rainforth, and Yarin Gal. Active Learning under Pool Set Distribution Shift and Noisy Data. *arXiv preprint*, 2021

Contributions. I was the lead author on the workshop submission, which focused on JEPIG. Tom Rainforth helped redraft the paper for the second arxiv submission which focused on JEPIG, and we jointly developed the EPIG term.

Prediction-Oriented Bayesian Active Learning [Smith et al., 2023]

> Freddie Bickford Smith*, Andreas Kirsch*, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. Prediction-Oriented Bayesian Active Learning. *International Conference on Artificial Intelligence and Statistics*, 2023

Contributions. Freddie and I are joint first-authors. I helped Freddie with the experiment design and provided feedback in discussion and encouragement throughout the project. The paper was significantly and majorly redrafted by Freddie, Tom, and Adam based on the earlier arXiv preprint.

Prioritized Training on Points that are Learnable, Worth Learning, and Not Yet Learnt [Mindermann et al., 2022]

Sören Mindermann*, Jan Markus Brauner*, Muhammed Razzak*, Mrinank Sharma*, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized Training on Points that are Learnable, Worth Learning, and not yet Learnt. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022

Contributions. While not being a joint first author, I contributed to the theory and helped draft the information theory subsection of the workshop submission. I wrote the comparison to EPIG in the appendix of the workshop submission. I independently developed the idea of a label-aware version of JEPIG before we realized that it was equivalent to what is now the unapproximated RHO-LOSS and joined the project.

Unifying Approaches in Active Learning and Active Sampling via Fisher Information and Information-Theoretic Quantities [Kirsch and Gal, 2022b]

Andreas Kirsch and Yarin Gal. Unifying Approaches in Active Learning and Active Sampling via Fisher Information and Information-Theoretic Quantities. *Transactions on Machine Learning Research*, 2022b

Contributions. I developed the paper and idea.

Black-Box Batch Active Learning for Regression [Kirsch, 2023a]

Andreas Kirsch. Black-Box Batch Active Learning for Regression. *Transactions on Machine Learning Research*, 2023a

Contributions. I am the single author of the paper.

Does "Deep Learning on a Data Diet" reproduce? Overall yes, but GraNd at Initialization does not [Kirsch, 2023b]

Andreas Kirsch. Does "Deep Learning on a Data Diet" reproduce? Overall yes, but GraNd at Initialization does not. *Transactions on Machine Learning Research*, 2023b

Contributions. I am the single author of the paper and received helpful feedback from the first author and senior author of the original paper, Paul et al. [2021].

A Note on "Assessing Generalization of SGD via Disagreement" [Kirsch and Gal, 2022a]

Andreas Kirsch and Yarin Gal. A Note on "Assessing Generalization of SGD via Disagreement". *Transactions on Machine Learning Research*, 2022a

Contributions. I developed the paper and idea.

Causal-BALD: Deep Bayesian Active Learning of Outcomes to Infer Treatment-Effects from Observational Data [Jesson et al., 2021]

Andrew Jesson*, Panagiotis Tigas*, Joost van Amersfoort, Andreas Kirsch, Uri Shalit, and Yarin Gal. Causal-BALD: Deep Bayesian Active Learning of Outcomes to Infer Treatment-Effects from Observational Data. In *Advances in Neural Information Processing Systems*, 2021

Contributions. While not being a joint first author, I contributed to the theory and proofs of ρ-BALD in particular.

Bibliography

Bayesian Inference of Individualized Treatment Effects using Multi-Task Gaussian Processes. In *Advances in Neural Information Processing Systems*, 2017.

Jacob D. Abernethy, Pranjal Awasthi, Matthäus Kleindessner, Jamie Morgenstern, Chris Russell, and Jie Zhang. Active Sampling for Min-Max Fairness. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

Jason Abrevaya, Yu-Chin Hsu, and Robert P Lieli. Estimating conditional average treatment effects. *Journal of Business & Economic Statistics*, 2015.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

Homayun Afrabandpey, Tomi Peltola, and Samuel Kaski. Human-in-the-loop Active Covariance Learning for Improving Prediction in Small Data Sets. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Ahmed M. Alaa and Mihaela van der Schaar. Bayesian Nonparametric Causal Inference: Information Rates and Learning Algorithms. *IEEE Journal of Selected Topics in Signal Processing*, 2018.

Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron C. Courville, and Yoshua Bengio. Variance Reduction in SGD by Distributed Importance Sampling. *arXiv preprint*, 2015.

Görkem Algan and Ilkay Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, 2021.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient Based Sample Selection for Online Continual Learning. *arXiv preprint*, 2019.

Ahsan S. Alvi, Bin Xin Ru, Jan-Peter Calliess, Stephen J. Roberts, and Michael A. Osborne. Asynchronous Batch Bayesian Optimisation with Improved Local Penalisation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Christos Anagnostopoulos, Fotis Savva, and Peter Triantafillou. Scalable Aggregation Predictive Analytics - A Query-Driven Machine Learning Approach. *Applied Intelligence*, 2018.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Róbert Ormándi, George E. Dahl, and Geoffrey E. Hinton. Large scale distributed neural network training through online distillation. In *International Conference on Learning Representations (ICLR)*, 2018.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common Voice: A Massively-Multilingual Speech Corpus. In *Proceedings of The Language Resources and Evaluation Conference (LREC)*, 2020.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

Jordan T. Ash and Ryan P. Adams. On Warm-Starting Neural Network Training. In *Advances in Neural Information Processing Systems*, 2020.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *International Conference on Learning Representations (ICLR)*, 2020.

Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Sham M. Kakade. Gone Fishing: Neural Active Learning with Fisher Embeddings. In *Advances in Neural Information Processing Systems*, 2021.

Parmida Atighehchian, Frédéric Branchaud-Charron, and Alexandre Lacoste. Bayesian active learning for production, a systematic study and a reusable library. *arXiv preprint*, 2020.

Les E. Atlas, David A. Cohn, and Richard E. Ladner. Training Connectionist Networks with Queries and Selective Sampling. In *Advances in Neural Information Processing Systems*, 1989.

Javad Azimi, Alan Fern, Xiaoli Zhang Fern, Glencora Borradaile, and Brent Heeringa. Batch Active Learning via Coordinated Matching. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

R. Harald Baayen and Rochelle Lieber. Word frequency distributions and lexical semantics. *Computational Humanities*, 1996.

Christina Baek, Yiding Jiang, Aditi Raghunathan, and Zico Kolter. Agreement-on-the-Line: Predicting the Performance of Neural Networks under Distribution Shift, 2022.

Rafael Ballester-Ripoll, Enrique G. Paredes, and Renato Pajarola. Sobol Tensor Trains for Global Sensitivity Analysis. *Reliability Engineering and System Safety*, 2019.

Tysam Balsam. hlb-CIFAR10, 2023. URL https://github.com/tysam-code/hlb-CIFAR10.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers, 2021.

Barber and Agakov. The IM algorithm: a variational approach to information maximization. *NeurIPS*, 2003.

Cenk Baykal, Lucas Liebenwein, Dan Feldman, and Daniela Rus. Low-Regret Active learning. *arXiv preprint*, 2021.

Beck and Arnold. *Parameter Estimation in Engineering and Science*. Wiley, 1977.

Belkin, Hsu, Ma, and Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences*, 2019.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, 2016.

William H. Beluch, Tim Genewein, Andreas Nürnberger, and Jan M. Köhler. The Power of Ensembles for Active Learning in Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

William Bialek and Naftali Tishby. Predictive information. *arXiv preprint*, 1999.

Sarah Bird, Miro Dudík, Richard Edgar, Brandon Horn, Roman Lutz, Vanessa Milan, Mehrnoosh Sameki, Hanna Wallach, and Kathleen Walker. Fairlearn: A toolkit for assessing and improving fairness in AI. Technical Report MSR-TR-2020-32, Microsoft, 2020.

Erdem Bıyık, Kenneth Wang, Nima Anari, and Dorsa Sadigh. Batch active learning using determinantal point processes. *arXiv preprint*, 2019.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems*, 2001.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Network. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho,

Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the Opportunities and Risks of Foundation Models. *arXiv preprint*, 2021.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via Bilevel Optimization for Continual Learning and Streaming. In *Advances in Neural Information Processing Systems*, 2020.

Zalán Borsos, Mojmír Mutný, Marco Tagliasacchi, and Andreas Krause. Data Summarization via Bilevel Optimization. *arXiv preprint*, 2021.

Léon Bottou and Yann LeCun. Large Scale Online Learning. In *Advances in Neural Information Processing Systems*, 2003.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Frédéric Branchaud-Charron, Parmida Atighehchian, Pau Rodríguez, Grace Abuhamad, and Alexandre Lacoste. Can Active Learning Preemptively Mitigate Fairness Issues? *ICLR Workshop on Responsible AI*, 2021.

Leo Breiman. Bagging Predictors. *Machine Learning*, 1996.

Leo Breiman. Random Forests. *Machine Learning*, 2001.

Klaus Brinker. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, 2020.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013.

Robert Burbidge, Jem J. Rowland, and Ross D. King. Active Learning for Regression Based on Query by Committee. In *Intelligent Data Engineering and Automated Learning*, 2007.

Daniel A Butts. How much information is associated with a particular stimulus? *Network: Computation in Neural Systems*, 2003.

Sylvain Calinon, Florent Guenter, and Aude Billard. On Learning, Representing, and Generalizing a Task in a Humanoid Robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 2007.

Colin Campbell, Nello Cristianini, and Alexander J. Smola. Query Learning with Large Margin Classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

Trevor Campbell and Tamara Broderick. Bayesian Coreset Construction via Greedy Iterative Geodesic Ascent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Trevor Campbell and Tamara Broderick. Automated Scalable Bayesian Inference via Hilbert Coresets. *Journal of Machine Learning Research (JMLR)*, 2019.

Daniel R. Cavagnaro, Jay I. Myung, Mark A. Pitt, and Janne V. Kujala. Adaptive Design Optimization: A Mutual Information-Based Approach to Model Discrimination in Cognitive Science. *Neural Computation*, 2010.

Chaloner and Verdinelli. Bayesian experimental design: a review. *Statistical Science*, 1995a.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 1995b.

Olivier Chapelle. Active Learning for Parzen Window Classifier. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2005.

Kamalika Chaudhuri, Sham M. Kakade, Praneeth Netrapalli, and Sujay Sanghavi. Convergence Rates of Active Learning for Maximum Likelihood Estimation. In *Advances in Neural Information Processing Systems*, 2015.

Jiefeng Chen, Frederick Liu, Besim Avci, Xi Wu, Yingyu Liang, and Somesh Jha. Detecting Errors and Estimating Accuracy on Unlabeled Data with Self-training Ensembles, 2021.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint*, 2017.

Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and Utilizing Deep Neural Networks Trained with Noisy Labels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways. *arXiv preprint*, 2022.

Sayak Ray Chowdhury and Aditya Gopalan. On Batch Bayesian Optimization. *arXiv preprint*, 2019.

Kenneth Ward Church and Patrick Hanks. Word Association Norms, Mutual Information and Lexicography. In *Annual Meeting of the Association for Computational Linguistics*, 1989.

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch Active Learning at Scale. In *Advances in Neural Information Processing Systems*, 2021.

Adam D. Cobb, Stephen J. Roberts, and Yarin Gal. Loss-Calibrated Approximate Inference in Bayesian Neural Networks. *arXiv preprint*, 2018.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*, 2017.

David A. Cohn. Neural Network Exploration Using Optimal Experiment Design. In *Advances in Neural Information Processing Systems*, 1993.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, 1996.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis,

Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via Proxy: Efficient Data Selection for Deep Learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian Process Optimization with Upper Confidence Bound and Pure Exploration. In *Machine Learning and Knowledge Discovery in Databases - European Conference*, 2013.

Cover and Thomas. *Elements of Information Theory*. John Wiley & Sons, 2005.

Thomas M Cover and A Thomas. Determinant inequalities via information theory. *SIAM Journal on Matrix Analysis and Applications*, 1988.

Pedram Daee, Tomi Peltola, Marta Soare, and Samuel Kaski. Knowledge elicitation via sequential probabilistic inference for high-dimensional prediction. *Machine Learning*, 2017.

Luke Nicholas Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. CINIC-10 is not ImageNet or CIFAR-10. *arXiv preprint*, 2018.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace Redux - Effortless Bayesian Deep Learning. In *Advances in Neural Information Processing Systems*, 2021.

Erik A. Daxberger and Bryan Kian Hsiang Low. Distributed Batch Gaussian Process Optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Tewodors Deneke, Habtegebreil Haile, Sébastien Lafond, and Johan Lilius. Video Transcoding Time Prediction for Proactive Load Balancing. In *International Conference on Multimedia and Expo*, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A large-scale hierarchical image database. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Kun Deng, Joelle Pineau, and Susan Murphy. Active Learning for Personalizing Treatment. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2011.

Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? Does it matter? *Structural Safety*, 2009.

Terrance DeVries and Graham W. Taylor. Learning Confidence for Out-of-Distribution Detection in Neural Networks. *arXiv preprint*, 2018.

Michael R DeWeese and Markus Meister. How to measure the information gained from one symbol. *Network: Computation in Neural Systems*, 1999.

Thomas G. Dietterich. Ensemble Methods in Machine Learning. In *Workshop of Multiple Classifier Systems at IEEE/CVF Proceedings*, 2000.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. In *Workshop Track Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Pinar Donmez and Jaime G. Carbonell. Optimizing estimated loss reduction for active sampling in rank learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. CatBoost: Gradient Boosting with Categorical Features Support. *arXiv preprint*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2020.

Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017. URL http://

archive.ics.uci.edu/ml.

Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-An Ma, Jasper Snoek, Katherine A. Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 2017.

Lewis P. G. Evans, Niall M. Adams, and Christoforos Anagnostopoulos. Estimating Optimal Active Learning via Model Retraining Improvement. *arXiv preprint*, 2015.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 2010.

Robert M. Fano. *Transmission of Information*. Wiley, 1962.

Sebastian Farquhar and Yarin Gal. What 'Out-of-distribution' Is and Is Not. *"ML Safety Workshop" Workshopat at NeurIPS*, December 2022.

Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On Statistical Bias In Active Learning: How and When to Fix It. In *International Conference on Learning Representations (ICLR)*, 2021.

Yassir Fathullah, Mark JF Gales, and Andrey Malinin. Ensemble distillation approaches for grammatical error correction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

Valerii Vadimovich Fedorov. *Theory of Optimal Experiments*. Elsevier, 1972.

Angelos Filos, Sebastian Farquhar, Aidan N. Gomez, Tim G. J. Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. A Systematic Comparison of Bayesian Deep Learning Robustness in Diabetic Retinopathy Tasks. *arXiv preprint*, 2019.

Louis Filstroff, Iiris Sundin, Petrus Mikkola, Aleksei Tiulpin, Juuso Kylmäoja, and Samuel Kaski. Targeted Active Learning for Bayesian Decision-Making. *arXiv preprint*, 2021.

Fisher. *Statistical Methods for Research Workers*. Oliver and Boyd, 1925.

Jose Pablo Folch, Robert M. Lee, Behrang Shafei, David Walz, Calvin Tsay, Mark van der Wilk, and Ruth Misener. Combining multi-fidelity modelling and asynchronous batch Bayesian Optimization. *Journal of Computers and Chemical Engineering*, 2023.

Edwin Fong and Chris C Holmes. On the marginal likelihood and cross-validation. *Biometrika*, 2020.

Adam Foster, Martin Jankowiak, Eli Bingham, Paul Horsfall, Yee Whye Teh, Tom Rainforth, and Noah D. Goodman. Variational Bayesian Optimal Experimental Design. In *Advances in Neural Information Processing Systems*, 2019.

Adam Foster, Desi R. Ivanova, Ilyas Malik, and Tom Rainforth. Deep Adaptive Design: Amortizing Sequential Bayesian Experimental Design. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

AE Foster. *Variational, Monte Carlo and policy-based approaches to Bayesian experimental design*. PhD thesis, University of Oxford, 2022.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Richard Fredlund, Richard M Everson, and Jonathan E Fieldsend. A Bayesian framework for active learning. In *International Joint Conference on Neural Networks (IJCNN)*, 2010.

Linton C Freeman. *Elementary applied statistics: for students in behavioral science*. John

Wiley & Sons, 1965.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective Sampling Using the Query by Committee Algorithm. *Machine Learning*, 1997.

Jerome H Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 1991.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016a.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016b.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian Active Learning with Image Data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Zijun Gao and Yanjun Han. Minimax Optimal Nonparametric Estimation of Heterogeneous Treatment Effects. In *Advances in Neural Information Processing Systems*, 2020.

Saurabh Garg, Sivaraman Balakrishnan, Zachary C. Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging Unlabeled Data to Predict Out-of-Distribution Performance, 2022.

Yonatan Geifman and Ran El-Yaniv. Deep Active Learning over the Long Tail. *arXiv preprint*, 2017.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 2007.

Daniel Golovin and Andreas Krause. Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization. *Journal of Artificial Intelligence Research*, 2011.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil D. Lawrence. Batch Bayesian Optimization via Local Penalization. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

Achintya Gopal. Why Calibration Error is Wrong Given Model Uncertainty: Using Posterior Predictive Checks with Deep Learning, 2021.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of Neural Networks by Enforcing Lipschitz Continuity. *Machine Learning*, 2021.

Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2D Image Registration in CT Images Using Radial Image Descriptors. In *Medical Image Computing and Computer-Assisted Intervention*, 2011.

Federica Granese, Marco Romanelli, Daniele Gorla, Catuscia Palamidessi, and Pablo Piantanida. DOCTOR: A Simple Method for Detecting Misclassification Errors, 2021.

Matthew J. Groves and Edward O. Pyzer-Knapp. Efficient and Scalable Batch Bayesian Optimization Using K-Means. *arXiv preprint*, 2018.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, 2017.

Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

Chengcheng Guo, B. Zhao, and Yanbing Bai. DeepCore: A Comprehensive Library for Coreset Selection in Deep Learning. *International Conference on Database and Expert Systems Applications*, 2022.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Yuhong Guo and Dale Schuurmans. Discriminative Batch Mode Active Learning. In *Advances in Neural Information Processing Systems*, 2007.

Jinyong Hahn. On the role of the propensity score in efficient semiparametric estimation of average treatment effects. *Econometrica*, 1998.

Ahmed Hammam, Frank Bonarens, Seyed Eghbal Ghobadi, and Christoph Stiller. Predictive Uncertainty Quantification of Deep Neural Networks using Dirichlet Distributions. In *Proceedings of the ACM Computer Science in Cars Symposium*, 2022.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.

Trevor Hastie, Jerome H. Friedman, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2001. ISBN 978-1-4899-0519-2.

Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian Deep Ensembles via the Neural Tangent Kernel. In *Advances in Neural Information Processing Systems*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of Tricks for Image Classification with Convolutional Neural Networks, 2018.

James J Heckman, Hidehiko Ichimura, and Petra E Todd. Matching as an econometric evaluation estimator: Evidence from evaluating a job training programme. *The Review of Economic Studies*, 1997.

James J Heckman, Hidehiko Ichimura, and Petra Todd. Matching as an econometric evaluation estimator. *The Review of Economic Studies*, 1998.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations (ICLR)*, 2019.

Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep Anomaly Detection with Outlier Exposure. In *International Conference on Learning Representations (ICLR)*, 2019.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural Adversarial Examples. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Philipp Hennig and Christian J. Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research (JMLR)*, 2012.

James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. Scalable Variational Gaussian Process Classification. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

Marek Herde, Zhixin Huang, Denis Huseljic, Daniel Kottke, Stephan Vogt, and Bernhard Sick. Fast Bayesian Updates for Deep Learning with a Use Case in Active Learning. *arXiv preprint*, 2022.

Daniel Hernández-Lobato, Jose Hernández-Lobato, and Pierre Dupont. Robust Multi-Class Gaussian Process Classification. *Advances in Neural Information Processing Systems*, 2011.

Daniel Hernández-Lobato, José Miguel Hernández-Lobato, Amar Shah, and Ryan P. Adams. Predictive Entropy Search for Multi-objective Bayesian Optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. In *Advances in Neural Information Processing Systems*, 2014.

José Miguel Hernández-Lobato, Michael A. Gelbart, Matthew W. Hoffman, Ryan P. Adams, and Zoubin Ghahramani. Predictive Entropy Search for Bayesian Optimization with Unknown Constraints. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Jennifer L Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 2011.

Geoffrey E. Hinton and Drew van Camp. Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights. In *Proceedings of the Annual ACM Conference on Computational Learning Theory (COLT)*, 1993.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint*, 2015.

Hjort, Holmes, Müller, and Walker. *Bayesian Nonparametrics*. Cambridge University Press, 2010.

Sepp Hochreiter and Jürgen Schmidhuber. Simplifying Neural Nets by Discovering Flat Minima. In *Advances in Neural Information Processing Systems*, 1994.

Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

Geoff Holmes, Eibe Frank, Dale Fletcher, and Corey Sterling. Efficiently correcting machine learning: considering the role of example ordering in human-in-the-loop training of image classification models. In *International Conference on Intelligent User Interfaces*, 2022.

David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart. A Framework and Benchmark for Deep Batch Active Learning for Regression. *arXiv preprint*, 2022.

Houlsby. *Efficient Bayesian active learning and matrix modelling*. PhD thesis, University of Cambridge, 2014.

Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Máté Lengyel. Bayesian Active Learning for Classification and Preference Learning. *arXiv preprint*, 2011.

Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized ODIN: Detecting Out-of-Distribution Image Without Learning From Out-of-Distribution Data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Haiwen Huang, Zhihan Li, Lulu Wang, Sishuo Chen, Xinyu Zhou, and Bin Dong. Feature Space Singularity for Out-of-Distribution Detection. In *Workshop on Artificial Intelligence Safety at AAAI Conference on Artificial Intelligence*, 2021.

Jiaji Huang, Rewon Child, Vinay Rao, Hairong Liu, Sanjeev Satheesh, and Adam Coates. Active Learning for Speech Recognition: the Power of Gradients. *arXiv preprint*, 2016.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *Advances in Neural Information Processing Systems*, 2019.

Yu Huang and Yue Chen. Autonomous Driving with Deep Learning: A Survey of State-of-Art Technologies. *arXiv preprint*, 2020.

Huszár. *Scoring rules, divergences and information in Bayesian machine learning*. PhD thesis, University of Cambridge, 2013.

Alexander Immer. Disentangling the Gauss-Newton Method and Approximate Inference for Neural Networks. *arXiv preprint*, 2020.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving Predictions of Bayesian Neural Nets via Local Linearization. In *The International Conference on Artificial Intelligence and Statistics*, 2021.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Rishabh K. Iyer, Ninad Khargoankar, Jeff A. Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory at Virtual Conference*, 2021.

David Janz, Jos van der Westhuizen, and José Miguel Hernández-Lobato. Actively Learning what makes a Discrete Sequence Valid. *arXiv preprint*, 2017.

Jeon. ThompsonBALD: a new approach to Bayesian batch active learning for deep learning via Thompson sampling. Master's thesis, University College London, 2020.

Andrew Jesson, Sören Mindermann, Uri Shalit, and Yarin Gal. Identifying Causal-Effect Inference Failure with Uncertainty-Aware Models. In *Advances in Neural Information Processing Systems*, 2020.

Andrew Jesson, Sören Mindermann, Yarin Gal, and Uri Shalit. Quantifying Ignorance in Individual-Level Causal-Effect Estimates under Hidden Confounding. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

Andrew Jesson, Panagiotis Tigas, Joost van Amersfoort, Andreas Kirsch, Uri Shalit, and Yarin Gal. Causal-BALD: Deep Bayesian Active Learning of Outcomes to Infer Treatment-Effects from Observational Data. In *Advances in Neural Information Processing Systems*, 2021.

Angela H. Jiang, Daniel L.-K. Wong, Giulio Zhou, David G. Andersen, Jeffrey Dean, Gregory R. Ganger, Gauri Joshi, Michael Kaminsky, Michael Kozuch, Zachary C. Lipton, and Padmanabhan Pillai. Accelerating Deep Learning by Focusing on the Biggest Losers. *arXiv preprint*, 2019.

Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J. Zico Kolter. Assessing Generalization of SGD via Disagreement. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022.

Tyler B. Johnson and Carlos Guestrin. Training Deep Models Faster with Robust, Approximate Importance Sampling. In *Advances in Neural Information Processing Systems*, 2018.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 1998.

Hlynur Jónsson, Giovanni Cherubini, and Evangelos Eleftheriou. Convergence Behavior of DNNs with Mutual-Information-Based Regularization. *Entropy*, 2020.

Keller Jordan. Calibrated Chaos: Variance Between Runs of Neural Network Training is Harmless and Inevitable. *arXiv preprint*, 2023.

Norman P. Jouppi, Cliff Young, Nishant Patil, David A. Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, 2017.

Cem Kalkanli and Ayfer Özgür. Batched Thompson Sampling. In *Advances in Neural Information Processing Systems*, 2021.

Nathan Kallus, Xiaojie Mao, and Angela Zhou. Interval Estimation of Individual-Level Causal Effects Under Unobserved Confounding. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised Bayesian Optimisation via Thompson Sampling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher D. Manning. Mind Your Outliers! Investigating the Negative Impact of Outliers on Active Learning for Visual Question Answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, 2021.

Angelos Katharopoulos and François Fleuret. Biased Importance Sampling for Deep Neural Network Training. *arXiv preprint*, 2017.

Angelos Katharopoulos and François Fleuret. Not All Samples Are Created Equal: Deep Learning with Importance Sampling. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Vishal Kaushal, Suraj Kothawade, Ganesh Ramakrishnan, Jeff A. Bilmes, and Rishabh Iyer. PRISM: A Unified Framework of Parameterized Submodular Information Measures for Targeted Data Subset Selection and Summarization. *arXiv preprint*, 2021.

Kenji Kawaguchi and Haihao Lu. Ordered SGD: A New Stochastic Optimization Framework for Empirical Risk Minimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Seho Kee, Enrique del Castillo, and George Runger. Query-by-committee Improvement with Diversity and Density in Batch Active Learning. *Information Sciences*, 2018.

Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems*, 2017.

Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. In *British Machine Vision Conference*, 2017.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations (ICLR)*, 2017.

Krishnateja Killamsetty, D. Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. GLISTER: Generalization based Data Subset Selection for Efficient and Robust Learning. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

KrishnaTeja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. GRAD-MATCH: Gradient Matching based Data Subset Selection for Efficient Deep Model Training. *arXiv preprint*, 2021a.

KrishnaTeja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh K. Iyer. GLISTER: Generalization based Data Subset Selection for Efficient and Robust Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021b.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised Learning with Deep Generative Models. In *Advances in Neural Information Processing Systems*, 2014.

Andreas Kirsch. Paper Review: Bayesian Model Selection, the Marginal Likelihood, and Generalization, 2022. URL https://blog.blackhc.net/2022/06/bayesian-model-selection-marginal-likehood-generalization/.

Andreas Kirsch. Black-Box Batch Active Learning for Regression. *Transactions on Machine Learning Research*, 2023a.

Andreas Kirsch. Does "Deep Learning on a Data Diet" reproduce? Overall yes, but GraNd at Initialization does not. *Transactions on Machine Learning Research*, 2023b.

Andreas Kirsch and Yarin Gal. A Practical & Unified Notation for Information-Theoretic Quantities in ML. *arXiv preprint*, 2021.

Andreas Kirsch and Yarin Gal. A Note on "Assessing Generalization of SGD via Disagreement". *Transactions on Machine Learning Research*, 2022a.

Andreas Kirsch and Yarin Gal. Unifying Approaches in Active Learning and Active Sampling via Fisher Information and Information-Theoretic Quantities. *Transactions on Machine Learning Research*, 2022b.

Andreas Kirsch, Clare Lyle, and Yarin Gal. Unpacking Information Bottlenecks: Unifying Information-Theoretic Objectives in Deep Learning. *arXiv preprint*, 2020.

Andreas Kirsch, Tom Rainforth, and Yarin Gal. Active Learning under Pool Set Distribution Shift and Noisy Data. *arXiv preprint*, 2021.

Andreas Kirsch, Jannik Kossen, and Yarin Gal. Marginal and Joint Cross-Entropies & Predictives for Online Bayesian Inference, Active Learning, and Active Sampling. *arXiv preprint*, 2022.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *Advances in Neural Information Processing Systems*, 2019.

Andreas Kirsch, Sebastian Farquhar, Parmida Atighehchian, Andrew Jesson, Frederic Branchaud-Charron, and Yarin Gal. Stochastic Batch Acquisition for Deep Active Learning. *Transactions on Machine Learning Research*, 2023.

Komaki. On asymptotic properties of predictive distributions. *Biometrika*, 1996.

Aran Komatsuzaki. One Epoch Is All You Need. *arXiv preprint*, 2019.

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic Beams and Where To Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Suraj Kothawade, Nathan Beck, KrishnaTeja Killamsetty, and Rishabh K. Iyer. SIMILAR:

Submodular Information Measures Based Active Learning In Realistic Scenarios. In *Advances in Neural Information Processing Systems*, 2021.

Suraj Kothawade, Vishal Kaushal, Ganesh Ramakrishnan, Jeff A. Bilmes, and Rishabh K. Iyer. PRISM: A Rich Class of Parameterized Submodular Information Measures for Guided Data Subset Selection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)*, 2008.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

Anders Krogh and Jesper Vedelsby. Neural Network Ensembles, Cross Validation, and Active Learning. In *Advances in Neural Information Processing Systems*, 1994.

Alex Kulesza and Ben Taskar. k-DPPs: Fixed-Size Determinantal Point Processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, 2019.

Alexandre Lacoste, Pau Rodríguez López, Frederic Branchaud-Charron, Parmida Atighehchian, Massimo Caccia, Issam Hadj Laradji, Alexandre Drouin, Matt Craddock, Laurent Charlin, and David Vázquez. Synbols: Probing Learning Algorithms with Synthetic Datasets. In *Advances in Neural Information Processing Systems*, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations (ICLR)*, 2020.

Leon Lang, Pierre Baudot, Rick Quax, and Patrick Forré. Information Decomposition Diagrams Applied beyond Shannon Entropy: A Generalization of Hu's Theorem. *arXiv preprint*, 2022.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Miguel Lázaro-Gredilla and Aníbal R Figueiras-Vidal. Marginalized Neural Network Mixtures for Large-Scale Regression. *IEEE Transactions on Neural Networks*, 2010.

Yann LeCun. The MNIST Database of Handwritten Digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 1998.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp.

In *Neural Networks: Tricks of the Trade - Second Edition*, 2012.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *"Challenges in Representation Learning" Workshop at ICML*, 2013.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. In *International Conference on Learning Representations (ICLR)*, 2018a.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Advances in Neural Information Processing Systems*, 2018b.

Kuang-Huei Lee, Ian Fischer, Anthony Liu, Yijie Guo, Honglak Lee, John Canny, and Sergio Guadarrama. Predictive Information Accelerates Learning in RL. In *Advances in Neural Information Processing Systems*, 2020.

Sun-Kyung Lee and Jong-Hwan Kim. BALD-VAE: Generative Active Learning based on the Uncertainties of Both Labeled and Unlabeled Data. In *International Conference on Robot Intelligence Technology and Applications (RiTA)*, 2019.

David D. Lewis and William A. Gale. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 1994.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Mingkun Li and Ishwar K. Sethi. Confidence-Based Active Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 1956.

Jeremiah Z. Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness. In *Advances in Neural Information Processing Systems*, 2020a.

Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based Out-of-distribution Detection. In *Advances in Neural Information Processing Systems*, 2020b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2022.

Fernando Llorente, Luca Martino, David Delgado, and Javier López-Santiago. Marginal Likelihood Computation for Model Selection and Hypothesis Testing: An Extensive Review. *SIAM Review*, 2023.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Quan Long. Multimodal information gain in Bayesian design of experiments. *Computational Statistics*, 2022.

Ilya Loshchilov and Frank Hutter. Online Batch Selection for Faster Training of Neural Networks. *arXiv preprint*, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International*

Conference on Learning Representations (ICLR), 2019.

Sanae Lotfi, Pavel Izmailov, Gregory W. Benton, Micah Goldblum, and Andrew Gordon Wilson. Bayesian Model Selection, the Marginal Likelihood, and Generalization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

Zhiming Luo, Frederic Branchaud-Charron, Carl Lemaire, Janusz Konrad, Shaozi Li, Akshaya Mishra, Andrew Achkar, Justin A. Eichel, and Pierre-Marc Jodoin. MIO-TCD: A New Benchmark Dataset for Vehicle Classification and Localization. *IEEE Transactions on Image Processing*, 2018.

Clare Lyle, Lisa Schut, Robin Ru, Yarin Gal, and Mark van der Wilk. A Bayesian Perspective on Training Speed and Model Selection. In *Advances in Neural Information Processing Systems*, 2020.

David J. C. MacKay. The Evidence Framework Applied to Classification Networks. *Neural Computation*, 1992a.

David J. C. MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 1992b.

David JC MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992c.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. A* Sampling. In *Advances in Neural Information Processing Systems*, 2014.

Wesley J. Maddox, Pavel Izmailov, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. A Simple Baseline for Bayesian Uncertainty in Deep Learning. In *Advances in Neural Information Processing Systems*, 2019.

Wesley J Maddox, Samuel Stanton, and Andrew G Wilson. Conditioning sparse variational gaussian processes for online decision-making. *Advances in Neural Information Processing Systems*, 2021.

Dhruv Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the Limits of Weakly Supervised Pretraining. In *Computer Vision - ECCV - European Conference, Proceedings, Part II*, 2018.

Andrey Malinin and Mark Gales. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. *Advances in Neural Information Processing Systems*, 2019.

Andrey Malinin and Mark J. F. Gales. Predictive Uncertainty Estimation via Prior Networks. In *Advances in Neural Information Processing Systems*, 2018.

Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. *arXiv preprint*, 2019.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An Empirical Model of Large-Batch Training. *arXiv preprint*, 2018.

William J. McGill. Multivariate information transmission. *Transactions of the IRE Professional Group on Information Theory*, 1954.

Sören Mindermann, Jan Markus Brauner, Muhammed Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized Training on Points that are Learnable, Worth Learning, and not yet Learnt. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

Thomas P. Minka. Bayesian model averaging is not model combination. 2002.

Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. Coresets for Data-efficient

Training of Machine Learning Models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Jonas Mockus. On Bayesian Methods for Seeking the Extremum. 1974.

Jishnu Mukhoti and Yarin Gal. Evaluating Bayesian Deep Learning Methods for Semantic Segmentation. *arXiv preprint*, 2018.

Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Prateek Munjal, Nasir Hayat, Munawar Hayat, Jamshid Sourati, and Shadab Khan. Towards Robust and Reproducible Active Learning using Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Kevin P. Murphy. *Machine learning - a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, 2012. ISBN 0262018020.

Chelsea Murray, James Urquhart Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Depth Uncertainty Networks for Active Learning. *arXiv preprint*, 2021.

Preetum Nakkiran and Yamini Bansal. Distributional Generalization: A New Kind of Generalization. *arXiv preprint*, 2020.

Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Do Deep Generative Models Know What They Don't Know? In *International Conference on Learning Representations (ICLR)*, 2019.

Radford M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, Canada, 1995.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 1978.

Mehdi Neshat, Bradley Alexander, Markus Wagner, and Yuanzhong Xia. A Detailed Comparison of Meta-Heuristic Methods for Optimising Wave Energy Converter Placements. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *"Deep Learning and Unsupervised Feature Learning" Workshop at NIPS*, 2011.

Jersey Neyman. Sur les Applications de la Théorie des Probabilités aux Experiences Agricoles: Essai des Principes. *Roczniki Nauk Rolniczych*, 1923.

Hieu T. Nguyen, Joseph Yadegar, Bailey Kong, and Hai Wei. Efficient Batch-Mode Active Learning of Random Forest. In *IEEE Statistical Signal Processing Workshop*, 2012.

Viet Cuong Nguyen, Wee Sun Lee, Nan Ye, Kian Ming Adam Chai, and Hai Leong Chieu. Active Learning for Probabilistic Hypotheses Using the Maximum Gibbs Error Criterion. In *Advances in Neural Information Processing Systems*, 2013.

Vu Nguyen, Santu Rana, Sunil Gupta, Cheng Li, and Svetha Venkatesh. Budgeted Batch Bayesian Optimization With Unknown Batch Sizes. *arXiv preprint*, 2017.

Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring Calibration in Deep Learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

Safiya Umoja Noble. *Algorithms of Oppression: How Search Engines Reinforce Racism*. New York University Press, 2018.

ChangYong Oh, Roberto Bondesan, Efstratios Gavves, and Max Welling. Batch Bayesian Optimization on Permutations using the Acquisition Weighted Kernel. *NeurIPS*, 2021.

Kazuki Osawa, Siddharth Swaroop, Mohammad Emtiyaz Khan, Anirudh Jain, Runa Eschenhagen, Richard E. Turner, and Rio Yokota. Practical Deep Learning with Bayesian Principles. In *Advances in Neural Information Processing Systems*, 2019.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized Prior Functions for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2018.

Ian Osband, Zheng Wen, Mohammad Asghari, Morteza Ibrahimi, Xiyuan Lu, and Benjamin Van Roy. Epistemic Neural Networks, 2021a.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Botao Hao, Morteza Ibrahimi, Dieterich Lawson, Xiuyuan Lu, Brendan O'Donoghue, and Benjamin Van Roy. The Neural Testbed: Evaluating Predictive Distributions, 2021b.

Ian Osband, Seyed Mohammad Asghari, Benjamin Van Roy, Nat McAleese, John Aslanides, and Geoffrey Irving. Fine-Tuning Language Models via Epistemic Neural Networks. *arXiv preprint*, 2022a.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Xiuyuan Lu, and Benjamin Van Roy. Evaluating high-order predictive distributions in deep learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2022b.

Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *Journal of ACM*, 2012.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, 2019.

Dongmin Park, Dimitris Papailiopoulos, and Kangwook Lee. Active Learning is a Strong Baseline for Data Subset Selection. In *"Has it Trained Yet?" at NeurIPS Workshop*, 2022.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch. In *"Autodiff" Workshop at NIPS*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 2019.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep Learning on a Data Diet: Finding Important Examples Early in Training. In *Advances in Neural Information Processing Systems*, 2021.

Tim Pearce, Alexandra Brintrup, and Jun Zhu. Understanding Softmax Confidence and Uncertainty. *arXiv preprint*, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research (JMLR)*, 2011a.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research (JMLR)*, 2011b.

Caroline Criado Perez. *Invisible Women: Exposing Data Bias in a World Designed for Men*. Random House, 2019.

Maya L Petersen, Kristin E Porter, Susan Gruber, Yue Wang, and Mark J Van Der Laan. Diagnosing and Responding to Violations in the Positivity Assumption. *Statistical Methods in Medical Research*, 2012.

Robert Pinsler, Jonathan Gordon, Eric T. Nalisnick, and José Miguel Hernández-Lobato. Bayesian Batch Active Learning as Sparse Subset Approximation. In *Advances in Neural Information Processing Systems*, 2019.

Geoff Pleiss, Tianyi Zhang, Ethan R. Elenberg, and Kilian Q. Weinberger. Identifying Mislabeled Data using the Area Under the Margin Ranking. In *Advances in Neural Information Processing Systems*, 2020.

Janis Postels, Hermann Blum, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. Quantifying Aleatoric and Epistemic Uncertainty Using Density Estimation in Latent Space. *arXiv preprint*, 2020.

L Prokhorenkova, G Gusev, A Vorobev, AV Dorogush, and A Gulin. CatBoost: Unbiased Boosting with Categorical Features. *arXiv preprint*, 2017.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research (JMLR)*, 2020.

Tom Rainforth, Robert Cornish, Hongseok Yang, and Andrew Warrington. On Nesting Monte Carlo Estimators. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Tom Rainforth, Adam Foster, Desi R. Ivanova, and Freddie Bickford Smith. Modern Bayesian Experimental Design. *arXiv preprint*, 2023.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining at KDD*, 2020.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised Learning with Ladder Networks. In *Advances in Neural Information Processing Systems*, 2015.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to Reweight Examples for Robust Deep Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A Survey of Deep Active Learning. *ACM Computing Surveys*, 2022.

James M. Robins, Miguel Angel Hernán, and Babette Brumback. Marginal Structural Models and Causal Inference in Epidemiology. *Epidemiology*, 2000.

David Rolnick, Andreas Veit, Serge J. Belongie, and Nir Shavit. Deep Learning is Robust to Massive Label Noise. *arXiv preprint*, 2017.

Nicholas Roy and Andrew McCallum. Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 1974.

Donald B Rubin. Randomization analysis of experimental data: The Fisher randomization test comment. *Journal of the American Statistical Association*, 1980.

Tim G. J. Rudner, Zonghao Chen, Yee Whye Teh, and Yarin Gal. Tractabe Function-Space Variational Inference in Bayesian Neural Networks. 2022.

Daniel Russo and Benjamin Van Roy. Learning to Optimize via Posterior Sampling. *Mathematics of Operations Research*, 2013.

Max Ryabinin, Andrey Malinin, and Mark Gales. Scaling ensemble distribution distillation to many classes with proxy targets. *Advances in Neural Information Processing Systems*, 2021.

Eman Saleh, Ahmad Tarawneh, M.Z. Naser, M. Abedi, and Ghassan Almasabha. You only design once (YODO): Gaussian Process-Batch Bayesian optimization framework for mixture design of ultra high performance concrete. *Construction and Building Materials*, 2022.

Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou. Explaining Aggregates for Exploratory Analytics. In *IEEE International Conference on Big Data*, 2018.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. In *International Conference on Learning Representations (ICLR)*, 2016.

Shira Schneider. *Algorithmic Bias: A New Age of Racism.* PhD thesis, 2021.

Greg Schohn and David Cohn. Less is More: Active Learning with Support Vector Machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

Jasjeet S Sekhon. The Neyman-Rubin model of causal inference and estimation via matching methods. *The Oxford Handbook of Political Methodology*, 2008.

Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations (ICLR)*, 2018.

Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian Process Regression: Active Data Selection and Test Point Rejection. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, 2000.

Burr Settles. Active Learning Literature Survey. *Machine Learning*, 2010.

Burr Settles and Mark Craven. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Conference on Empirical Methods in Natural Language Processing*, 2008.

Burr Settles, Mark Craven, and Soumya Ray. Multiple-Instance Active Learning. In *Advances in Neural Information Processing Systems*, 2007.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by Committee. In *Proceedings of the Annual Conference on Computational Learning Theory (COLT)*, 1992.

Amar Shah and Zoubin Ghahramani. Parallel Predictive Entropy Search for Batch Global Optimization of Expensive Objective Functions. In *Advances in Neural Information Processing Systems*, 2015.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of IEEE*, 2016.

Mohammad Hossein Shaker and Eyke Hüllermeier. Aleatoric and Epistemic Uncertainty with Random Forests. In *Advances in Intelligent Data Analysis - Proceedings of the International Symposium on Intelligent Data Analysis*, 2020.

Uri Shalit, Fredrik D. Johansson, and David A. Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*,

1948.

Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 2003.

Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do Bayesian Neural Networks Need To Be Fully Stochastic? *arXiv preprint*, 2022.

John Shawe-Taylor, Nello Cristianini, et al. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep Active Learning for Named Entity Recognition. In *International Conference on Learning Representations (ICLR)*, 2018.

Claudia Shi, David M. Blei, and Victor Veitch. Adapting Neural Networks for the Estimation of Treatment Effects. In *Advances in Neural Information Processing Systems*, 2019.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv preprint*, 2019.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the Black Box of Deep Neural Networks via Information. *arXiv preprint*, 2017.

Aditya Siddhant and Zachary C. Lipton. Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

Shoaib Ahmed Siddiqui, Nitarshan Rajkumar, Tegan Maharaj, David Krueger, and Sara Hooker. Metadata Archaeology: Unearthing Data Subsets by Leveraging Training Dynamics. In *International Conference on Learning Representations (ICLR)*, 2023.

Eero Siivola, Akash Kumar Dhaka, Michael Riis Andersen, Javier González, Pablo Garcia Moreno, and Aki Vehtari. Preferential Batch Bayesian Optimization. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2021.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational Adversarial Active Learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Dominik Ślęzak, Marek Grzegorowski, Andrzej Janusz, Michał Kozielski, Sinh Hoa Nguyen, Marek Sikora, Sebastian Stawicki, and Łukasz Wróbel. A Framework for Learning and Embedding Multi-Sensor Forecasting Models into a Decision Support System: A Case Study of Methane Concentration in Coal Mines. *Information Sciences*, 2018.

Lewis Smith and Yarin Gal. Understanding Measures of Uncertainty for Adversarial Example Detection. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2018.

Lewis Smith, Joost van Amersfoort, Haiwen Huang, Stephen J. Roberts, and Yarin Gal. Can convolutional ResNets approximately preserve input distances? A frequency analysis perspective. *arXiv preprint*, 2021.

Samuel L. Smith and Quoc V. Le. A Bayesian Perspective on Generalization and Stochastic Gradient Descent. In *International Conference on Learning Representations (ICLR)*, 2018.

Freddie Bickford Smith, Andreas Kirsch, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. Prediction-Oriented Bayesian Active Learning. *International Conference on Artificial Intelligence and Statistics*, 2023.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems*, 2005.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, 2012.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. 2010.

Cathie Sudlow, John Gallacher, Naomi Allen, Valerie Beral, Paul Burton, John Danesh, Paul Downey, Paul Elliott, Jane Green, Martin Landray, et al. UK Biobank: An Open Access Resource for Identifying the Causes of a Wide Range of Complex Diseases of Middle and Old Age. *Plos Medicine*, 2015.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Shengyang Sun, Guodong Zhang, Chaoqi Wang, Wenyuan Zeng, Jiaman Li, and Roger B. Grosse. Differentiable Compositional Kernel Learning for Gaussian Processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Shengyang Sun, Daniele Calandriello, Huiyi Hu, Ang Li, and Michalis K. Titsias. Information-theoretic Online Memory Selection for Continual Learning. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022.

Iiris Sundin, Tomi Peltola, Luana Micallef, Homayun Afrabandpey, Marta Soare, Muntasir Mamun Majumder, Pedram Daee, Chen He, Baris Serim, Aki S. Havulinna, Caroline Heckman, Giulio Jacucci, Pekka Marttinen, and Samuel Kaski. Improving genomics-based predictions for precision medicine through active elicitation of expert knowledge. *Bioinformatics*, 2018.

Iiris Sundin, Peter Schulam, Eero Siivola, Aki Vehtari, Suchi Saria, and Samuel Kaski. Active Learning for Decision-Making from Imbalanced Observational Data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Wei Tan, Lan Du, and Wray L. Buntine. Diversity Enhanced Active Learning with Strictly Proper Scoring Rules. In *Advances in Neural Information Processing Systems*, 2021.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

Lu Tian, Ash A. Alizadeh, Andrew J. Gentles, and Robert Tibshirani. A Simple Method for Estimating Interactions between a Treatment and a Large Number of Covariates. *Journal of the American Statistical Association*, 2014.

Yonglong Tian, Olivier J. Hénaff, and Aäron van den Oord. Divide and Contrast: Self-supervised Learning from Uncurated Data. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Panagiotis Tigas, Yashas Annadani, Andrew Jesson, Bernhard Schölkopf, Yarin Gal, and Stefan Bauer. Interventions, Where and How? Experimental Design for Causal Models at Scale. *arXiv preprint*, 2022.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An Empirical Study of Example Forgetting during Deep Neural Network Learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Simon Tong. *Active learning: theory and applications*. PhD thesis, Stanford University, 2001.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2020.

Dustin Tran, Jeremiah Z. Liu, Michael W. Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda Mariet, Huiyi Hu, Neil Band, Tim G. J. Rudner, Karan Singhal, Zachary Nado, Joost van Amersfoort, Andreas Kirsch, Rodolphe Jenatton, Nithum

Thain, Honglin Yuan, Kelly Buchanan, Kevin Murphy, D. Sculley, Yarin Gal, Zoubin Ghahramani, Jasper Snoek, and Balaji Lakshminarayanan. Plex: Towards Reliability using Pretrained Large Model Extensions. *arXiv preprint*, 2022.

Toan Tran, Thanh-Toan Do, Ian D. Reid, and Gustavo Carneiro. Bayesian Generative Active Deep Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Joost van Amersfoort. Minimal CIFAR-10, 2021. URL https://github.com/y0ast/pytorch-snippets/tree/main/minimal_cifar.

Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty Estimation Using a Single Deep Deterministic Neural Network. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression. *arXiv preprint*, 2021.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luís Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 2013.

Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data, Second Editiontion*. Springer, 2006. ISBN 978-0-387-30865-4.

Sahil Verma and Julia Rubin. Fairness definitions explained. In *Proceedings of the International Workshop on Software Fairness, FairWare@ICSE*, 2018.

Julien Villemonteix, Emmanuel Vázquez, and Eric Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 2009.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations (ICLR)*, 2019.

Chaoqi Wang, Shengyang Sun, and Roger B. Grosse. Beyond Marginal Uncertainty: How Accurately can Bayesian Regression Models Estimate Posterior Predictive Correlations? In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-Effective Active Learning for Deep Image Classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Zheng Wen, Ian Osband, Chao Qin, Xiuyuan Lu, Morteza Ibrahimi, Vikranth Dwaracherla, Mohammad Asghari, and Benjamin Van Roy. From Predictions to Decisions: The Importance of Joint Predictive Distributions, 2021.

Ross Wightman. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models, 2019.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Paul L Williams. *Information dynamics: Its theory and application to embodied cognitive systems*. PhD thesis, PhD thesis, Indiana University, 2011.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. In *Advances in Neural Information Processing Systems*, 2020.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep Kernel Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*,

2016.

Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R. Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, A. Taylan Cemgil, S. M. Ali Eslami, and Olaf Ronneberger. Contrastive Training for Improved Out-of-Distribution Detection. *arXiv preprint*, 2020.

Guoxuan Xia and Christos-Savvas Bouganis. On the Usefulness of Deep Ensemble Diversity for Out-of-Distribution Detection. *arXiv preprint*, 2022.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint*, 2017.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Yu Xie, Jennie E. Brand, and Ben Jann. Estimating Heterogeneous Treatment Effects with Observational Data. *Sociological Methodology*, 2012.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A Theory of Usable Information under Computational Constraints. In *International Conference on Learning Representations (ICLR)*, 2020.

Chhavi Yadav and Léon Bottou. Cold Case: The Lost MNIST Digits. In *Advances in Neural Information Processing Systems*, 2019.

David Yarowsky. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Annual Meeting of the Association for Computational Linguistics*, 1995.

Raymond W Yeung. A new outlook on Shannon's information measures. *IEEE Transactions on Information Theory*, 1991.

Raymond W Yeung. *Information Theory and Network Coding*. Information Technology: Transmission, Processing and Storage. Springer US, 2008. ISBN 9780387792347.

Kun Yi and Jianxin Wu. Probabilistic End-To-End Noise Correction for Learning With Noisy Labels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Kai Yu, Jinbo Bi, and Volker Tresp. Active learning via transductive experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference*, 2016.

Xueying Zhan, Qingzhong Wang, Kuan-Hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B. Chan. A Comparative Survey of Deep Active Learning. *arXiv preprint*, 2022a.

Xueying Zhan, Yaowei Wang, and Antoni B Chan. Asymptotic optimality for active learning processes. In *Uncertainty in Artificial Intelligence*, 2022b.

Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger B. Grosse. Noisy Natural Gradient as Variational Inference. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Jiong Zhang, Hsiang-Fu Yu, and Inderjit S. Dhillon. AutoAssist: A Framework to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 2019a.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019b.

Min-Ling Zhang and Zhi-Hua Zhou. Exploiting Unlabeled Data to Enhance Ensemble Diversity. *Data Mining and Knowledge Discovery*, 2013.

Guang Zhao, Edward R. Dougherty, Byung-Jun Yoon, Francis J. Alexander, and Xiaoning Qian. Bayesian Active Learning by Soft Mean Objective Cost of Uncertainty. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021a.

Guang Zhao, Edward R. Dougherty, Byung-Jun Yoon, Francis J. Alexander, and Xiaoning Qian. Efficient Active Learning for Gaussian Process Classification by Error Reduction. In *Advances in Neural Information Processing Systems*, 2021b.

Guang Zhao, Edward R. Dougherty, Byung-Jun Yoon, Francis J. Alexander, and Xiaoning Qian. Uncertainty-aware Active Learning for Optimal Bayesian Classifier. In *International Conference on Learning Representations (ICLR)*, 2021c.

Xiao Zhou, Renjie Pi, Weizhong Zhang, Yong Lin, Zonghao Chen, and T. Zhang. Probabilistic Bilevel Coreset Selection. *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

Zhu, Lafferty, and Ghahramani. Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.