Uncertainty Estimation: Single Forward Pass Methods and Applications in Active Learning

Joost René van Amersfoort

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022

Acknowledgements

I am very fortunate to have been part of two great labs: OATML and OxCSML. When I joined OATML as part of its first few DPhils students, we had the opportunity to craft a new lab that we truly enjoined. We tried out many talk formats, compute setups, collaborations, office arrangements, and research directions. I have learned so much building all of this up with the rest of OATML, and I am very happy that I was there for the beginning. OxCSML provided a welcome stable environment, with tried and tested talk formats and long-standing research directions. A comforting alternative to OATML's constant change.

Yarin, I am impressed with what you have built up with OATML over just a few years. Thank you for taking me along on this ride. Our many conversations were always a source of energy and inspiration. My favourite moments were when I was convinced your reply did not make sense. To only later realise, that you *did* understand me and were already some steps ahead.

Yee Whye, I continue to be amazed by the number of things you are involved in. You manage to have an impact on such a wide array of projects, with the special highlight being COVID-19 modelling at SAGE. Your breadth of knowledge is a real inspiration, and somehow you always ask just the right question at the right moment.

At OATML, no one had more impact on my research than Lewis Smith. Our conversations have made this thesis possible. Sebastian Farquhar, Milad Alizadeh, Andreas Kirsch, Andrew Jesson, Jishnu Mukhoti, Panagiotis Tigas, Pascal Notin, Aidan Gomez, Oscar Key, Chetan Gohil, Bas Veeling, Adam Foster, Emilien Dupont, Haiwen Huang, and John Ryan: I had a lot of fun working with each and every one of you. You all taught me something different, and I am curious to see where you will go. Angelos Filos, Clare Lyle, Freddie Bickford Smith, Gunshi Gupta, Jan Brauner, Jannik Kossen, Kelsey Doerksen, Lisa Schut, Lorenz Kuhn, Sören Mindermann, and Tim Rudner: thank you for the fun discussions and making my time in Oxford so enjoyable. A special thank you to Ian Collier of the IT team at the Computer Science department–your work has made a big difference. I hope OATML remains the productive and high energy place for a long time to come!

Korfball was the heart of my non-academic life at Oxford. Thank you to Sophie, Joe, Matt, Amy, Stephan, Niall and all other Korf friends for providing many hours of fun, four won varsities, and one league championship. Kara Allum, you have influenced my view on life more than you know. I hope that we remain friends for a long time.

Gratitude to the following Oxford establishments from one happy customer: ATS, Najar's, Aleppo's, Opera, Hamblin, Tse Noodle, Za'atar, Zhang Ji, The Chester Arms, Oli's Thai, Peppers, The Greek Takeaway, Currydor, Brew, Missing Bean, 101 Coffee, The Victoria, The Star, The Cape of Good Hope, The Fir Tree, and The Royal Oak.

My DPhil would not have been the same, or even materialised really, without Luisa Zintgraf. Your help and encouragement have made the difference during so many moments. We survived moving to a new country, a pandemic, and two DPhils, I am sure we are ready for everything that comes next. Thank you for being my source of chaos and love.

I want to thank my family for all their support, (surprise) visits, packages, and phone calls. My sisters Emma and Irene, my parents Wilma and Steef, you are all wonderful. Thank you to my extended family Chiara, Gabriele, Deborah, and Reinhold, I look forward to hosting you many more times.

Kyriacos, Eva, Jonas and Panos, thank you for keeping me cool and fun in Oxford. To my Dutch(-based) friends, Thijmen, Lukas, Pascal, Otto, Yvonne, Hüseyin, Bas, Thijs, Marcel, Joost–"Doorn", thank you for support and visits. I look forward to being around more often!

It is safe to say that this thesis was not just a labour of my love, but also of the many around me. I plan to pay their support forward, and be the rock that others were for me.

Princes Street, Oxford, June 2022

Abstract

Machine Learning (ML) models are now powerful enough to be used in complex automated decision-making settings such as autonomous driving and medical diagnosis. Despite being very accurate in general, these models do still make mistakes. A critical factor in being able to depend on such models is that they can quantify the *uncertainty* of their predictions, and it is paramount that this is taken into account by users of the model. Unfortunately, deep learning models cannot readily express their uncertainty, rendering them unsafe for many real-world applications.

Bayesian modelling provides a mathematical framework for learning models that can express their uncertainty. However, exact Bayesian methods are computationally expensive to learn and evaluate, and approximate methods often reduce accuracy or are still prohibitively expensive. Meanwhile, ML models continue to increase in number of parameters, meaning that one has to make a decision between being (more) Bayesian or using a larger model. So far it has always fallen in favour of larger models.

Instead of building on Bayesian methods, we deconstruct uncertainty estimation and formulate desiderata that we base our work on throughout the thesis (Chapter 1). In Chapter 3, we introduce a new model (DUQ) that is able to estimate uncertainty in a single forward pass by carefully constructing the model's parameter and output space based on the desiderata. We then extend this model in Chapter 4 (DUE) by placing it in the framework provided by Deep Kernel Learning. This enables the model to work well for both classification and regression tasks (as opposed to just classification), and estimate uncertainty over a batch of inputs jointly. Both models are competitive with standard softmax models in terms of accuracy and speed, while having significantly improved uncertainty estimation.

We additionally consider the problem of Active Learning (AL), where the goal is to maximise label efficiency by selecting only the most informative data points to be labelled. In Section 4.5, we evaluate the DUE model in AL for personalised healthcare. Here, the labelled dataset needs to adhere to specific assumptions made in causal inference, which makes this a challenging problem. In Chapter 5, we look at AL in the *batch* setting. We show that current methods do not select diverse batches of data, and we introduce a principled method to overcome this issue.

Building upon deep kernel learning, this thesis provides a compelling foundation for *single forward pass uncertainty* and advances the state of the art in active learning. In the conclusions (Section 6, and at the end of each chapter), we discuss how users of ML models could make use of these tools for making sound and confident decisions.

Contents

1 Introduction

Consider the question: "what will the weather be like today?". When formulating a reply, which involves making a prediction about the future, one can consider many sources of information: the season in one's current location, a forecast made by a meteorologist, one's gut feeling, or the clouds in the sky. A combination of a sunny forecast and dark clouds leads to a tricky situation: usually the forecast is right, but the sky indicates otherwise. In this case, a reasonable prediction could be: "it will mostly be sunny today, but it might rain a bit first". Based on this answer, a cyclist might make the decision to bring a light, rainproof jacket, but not cancel their trip.

The uncertainty of the given answer is a fundamental component of the decision-making that depends on it. If uncertainty is low, then we can act confidently based on it. If uncertainty is high, then it might be necessary to collect more information and make new predictions, or alternatively take extra precautions.

Such predictions can be made not only by humans, but we are now entering an era where machine learning (ML) models can do so as well. These are now powerful enough to make useful predictions on many complex tasks. When integrating their predictions into any subsequent decision-making, it is crucial that the model's uncertainty is quantified accurately. Overconfident but incorrect predictions could have catastrophic consequences, while staying too cautious would diminish the value of the predictions.

ML models can be created by learning from large sets of data, called the training data. Their predictive performance is evaluated on a held-out dataset, the test set, and when the performance is deemed sufficient, the model can be deployed. With the advent of deep learning, made possible by the increased availability of fast computers and large amounts of data, the complexity of tasks that a model can be learned for has increased substantially. As a result, there are now more and higher stake situations where ML models are able to make useful predictions. In some cases, ML models can even be relied upon for automated decision-making, translating predictions into actions in an automated fashion, for example in autonomous driving or medical diagnosis. As the stakes are higher, it has become even more important for these models to quantify their uncertainty correctly.

Uncertainty in learned models can come from a range of sources. For example, the data at deployment could be presented differently from the training data, or it could contain elements not seen in the training data. Some sources of uncertainty can be mitigated, for instance one could improve the diversity and size of the training dataset and use it to learn a new model. Others are simply inherent, for example if there is insufficient information in the input to make a prediction: imagine counting cows in a satellite image where the fields are obscured by clouds. A natural way to express uncertainty is to use probabilistic modelling, where the model predicts a distribution over possible outcomes, e.g., this image contains a cat with probability 80% and a dog with probability 20%, but a horse with 0% probability. In that setting, confidence can be determined by looking at the probability that the model assigns to each outcome. It is important that the estimated confidence is calibrated such that it can be evaluated properly in any subsequent decision-making. 50% uncertainty should indeed mean that across many predictions the model is correct half the time.

A mathematical framework for learning models that can express their uncertainty is Bayesian modelling (MacKay, 1992; Neal, 1996; Ghahramani, 2015). In Bayesian modelling, model parameters are represented using distributions and updated using Bayes' rule. Predictions are made by integrating over all possible parameter

settings and when those different parameter settings lead to a wide variety of outcomes then uncertainty is high, but if the different settings agree on a prediction then uncertainty is low. While Bayesian modelling is an excellent way to obtain uncertainty estimates, it does come at increased computational cost compared to learning a deterministic model. Exact Bayesian inference in models without a closed form predictive distribution (such as neural networks) is intractable. Sampling procedures, such as the Hamiltonian Monte Carlo algorithm (Neal, 1996), for models with more than a handful of parameters are very computationally intensive. Other approximate methods still incur a significant extra computaional cost or reduce accuracy. We discuss why Bayesian models can be computationally prohibitive to learn and look at several approximations in Chapter 2. Meanwhile, it has become clear that increasing the number of parameters in a model and training them on more data leads to increased accuracy (Kaplan et al., 2020). This creates the situation where one has to trade off predictive performance with being Bayesian. So far, the trade-off has been decided in favour of making models larger, rather than making them (more) Bayesian Despite Bayesian methods providing more benefit to the user when there are more parameters and the risk of overfitting (and overconfidence) increases. This raises the question: are there alternative methods for estimating uncertainty in deep learning that do not require such a trade-off?

In the first part of this thesis, we look for ways of obtaining uncertainty estimates that do not reduce accuracy or incur a large computational cost. We start by deconstructing what it means to estimate uncertainty and formulate two sets of desiderata: one set that an uncertainty estimation method *must* meet for the output of the method to be considered an uncertainty estimate and a second set that are required only for certain applications, such medical diagnostics. Subsequently, we look for mechanisms that enable current models to fulfil these desiderata while only incurring a minimal increase in computational cost. These mechanisms include adaptations to current deep learning models, but also alternative ways of quantifying uncertainty.

In the second part of this thesis we consider the problem setting of Bayesian active learning. The goal of active learning (AL) is to acquire labels only for the most informative data points and thereby minimise the number of labels necessary for obtaining a well performing model. In practice, AL is an iterative process consisting of two steps. The first step is to train a model on the currently available labelled training data and subsequently evaluated. The process is stopped if the performance of the trained model is deemed sufficient or the labelling budget is spent. The second step is to use the trained model to select the most informative new data points to label. In *Bayesian* active learning uncertainty is used as a measure of informativeness. We use Bayesian active learning as a way to evaluate uncertainty estimation, and also suggest new methods for it.

1.1 Deconstructing Uncertainty Estimation

If we want to devise new methods for uncertainty estimation that are not necessarily Bayesian, we first need to establish what is meant by estimating uncertainty. We therefore list three desiderata that, when followed, should lead to a value that can intuitively be interpreted as uncertainty. Note that adherence to these desiderata will never be perfect, nevertheless we prefer models that are closer to adhering them, for example as measured by benchmarks on large sets of data.

1. When the model predicts with low uncertainty, the prediction is expected to be accurate. While high uncertainty should be reserved for predictions that are closer to a random guess. This is known as calibration (discussed in Section 2.3.2).

2. The model should signal high uncertainty for data points that *cannot* be classified correctly, either because the data point does not belong to any of the classes seen during training or it is so different from the training data that the model cannot generalise to it.

3. If the training and test data are independently and identically distributed (iid), then adding additional training data means uncertainty on the test set is generally expected to go down.

We have separated Point 1 and Point 2 into two desiderata despite being intuitively very similar, because they require different evaluation methods, which we discuss in Section 2.3. Point 3 describes the relation between size of the dataset and uncertainty: more data is usually expected to reduce the model's uncertainty.

Related to these desiderata are the concepts of epistemic and aleatoric uncertainty, which are the two main sources of uncertainty. Epistemic uncertainty arises when an input data point at test or evaluation time cannot be generalised to from the training data. Alternatively put, epistemic uncertainty is uncertainty over which solution is correct given limited information. This uncertainty is often represented by a distribution over model parameters. If we were to add data points similar to that input point to the training data and retrain the model, then this uncertainty is expected to decrease. An example of epistemic uncertainty is a data point from a class that only has a few examples in the training dataset, and that the model cannot generalise to yet. Aleatoric uncertainty arises when an input data point contains too much noise or is too ambiguous to be classified confidently. Even if we were to add similar data points to the training set, we would not be able to reduce this uncertainty. An example of aleatoric uncertainty is a handwritten digit that looks like both a "1" and a "7". This means the digit does not actually belong to either class and its true label is noisy. Epistemic and aleatoric uncertainty are also sometimes called reducible and irreducible uncertainty. For some applications, it is important that the uncertainty can be ascribed to a particular source: in active learning we are interested only in epistemically uncertain points for acquiring new labels. Labelling an ambiguous data point (with high aleatoric uncertainty) would not improve the model. Additional details and examples of these sources of uncertainty are given in Section 2.3.1.

There are four additional desiderata that are required in some applications of uncertainty estimation, but not expected of all methods in general:

1. If the model includes a large neural network, then it can be prohibitively expensive if the method requires multiple forward passes. For example, in applications where decisions need to be made in real-time, or where multiple forward passes would make the cost of the computation prohibitive for its intended use case. In these cases, it should only be necessary to do a single forward pass.

2. It can be necessary to be able to disentangle the source of uncertainty: aleatoric or epistemic. As mentioned previously, this is for example the case in active learning. Additionally, when a model signals a constant high amount of aleatoric uncertainty this could mean something is wrong with the way the data is collected, for instance the lens could be dirty. While constant high epistemic uncertainty implies the model and/or the training data are not sufficient for the task.

3. It is useful to be able to compute uncertainty on a set of data points ("joint uncertainty"). One application where this is important is batch active learning, where multiple informative points are selected to label at the same time.

4. Generally, uncertainty should remain high on unseen parts of the input domain, even when the model is overfit. In many applications, such as medical diagnostics, it is important the model reverts to a baseline level of uncertainty away from the training data. We discuss this behaviour in more detail in Section 2.3.5.

Research Questions We now formulate several concrete research questions based on the desiderata. We start by considering how to evaluate if a model satisfies the desiderata and verify to what extent current models do so. Evaluation methods are discussed in Section 2.3, and current methods are described in Section 2.2. Subsequently, we focus on the question of how to estimate uncertainty in a single forward pass. It is known that standard softmax models can suffer from spurious confident extrapolation (Gal, 2016). However, it is unclear if it is possible to adapt

softmax models to avoid this behaviour, and we study this question in Chapter 3. We then consider the question if it is possible to design a model that is able to satisfy all four additional desiderata. We look at this question in Chapter 4 and use the insights from Chapter 3. In the second part of this thesis, we look at the desideratum of joint uncertainty through the lens of batch AL. We research the open question for a tractable estimator of this uncertainty that is practically useful for AL in Chapter 5.

1.2 Contributions

The contributions of this thesis are split into three parts: uncertainty estimation in a single forward pass, uncertainty estimation in deep kernel learning, and uncertainty estimation for active learning. The source code for the models and experiments for all chapters is available under a free and open source licence and linked in the introduction of each chapter.

Uncertainty Estimation in a Single Forward Pass In Chapter 3, we propose a new model that is able to obtain uncertainty estimates on high dimensional inputs in a single forward pass. We do this by placing restrictions on the parameters of the model and use a distance-based model output. The uncertainty estimates match or outperform Deep Ensembles (Lakshminarayanan et al., 2017) in high dimensions. In low dimensions, the proposed model remains uncertain even far away from the training data, while the Deep Ensemble elements concentrate on a single solution and are certain everywhere except on the decision boundary. The chapter is based in large part on van Amersfoort, Smith, Teh, and Gal (2020), which was published at ICML 2020, and contains a discussion of a simplified alternative by Mukhoti*, Kirsch*, van Amersfoort, Torr, and Gal (2023), published at CVPR 2023.

Uncertainty in Deep Kernel Learning In Chapter 4, we investigate uncertainty estimation in Deep Kernel Learning (DKL) (Hinton et al., 2008; Calandra et al., 2016; Wilson et al., 2016a), and identify several shortcomings of previous DKL models for which we provide a practical solution. The model introduced in this chapter

satisfies all presented desiderata. This chapter is related to the previous chapter in that it places a similar restriction on the parameters to improve uncertainty estimation in DKL. The difference is that the model introduced in this chapter works on classification and regression (compared to just classification) and uses an inducing point Gaussian process output. This chapter is based on van Amersfoort, Smith, Jesson, Key, and Gal (2021), which was presented at the Bayesian Deep Learning workshop at NeurIPS 2021. In Section 4.5, we additionally apply the model introduced in this chapter on active learning for causal inference based on Jesson*, Tigas*, van Amersfoort, Kirsch, Shalit, and Gal (2021), published at NeurIPS 2021.

Batch Active Learning In Chapter 5, we look at Bayesian active learning in the batch acquisition setting, where we use uncertainty to select batches of data to label. We identify a pathology with the application of current acquisition functions in the batch active learning setting. We subsequently derive a principled alternative and show its efficacy in a range of scenarios. This chapter is based on Kirsch*, van Amersfoort*, and Gal (2019), which was published at NeurIPS 2019. We additionally include a section on using large pre-trained models for active learning. We estimate informativeness of individual data points with these models in a single forward pass. This section is based Tran, Liu, Dusenberry, Phan, Collier, Ren, Han, Wang, Mariet, Hu, Band, Rudner, Nado, van Amersfoort, Kirsch, Jenatton, Thain, Buchanan, Murphy, Sculley, Gal, Ghahramani, Snoek, and Lakshminarayanan (2022) which is presented at two workshops: “Pre-training: Perspectives, Pitfalls, and Paths Forward” and “Principles of Distribution Shift” at ICML 2022, and in submission to JMLR. This work was done in collaboration with Google Brain.

2 Background

Contents

This chapter discusses existing methods and definitions in Bayesian deep learning, uncertainty estimation, and active learning. We derive several commonly used models and approximate inference techniques, such as variational inference. We also review ways to evaluate uncertainty estimation methods and disambiguate terms used in the literature. This chapter is restricted to background knowledge relevant for multiple chapters in this thesis, and leaves specific background knowledge and related work to each chapter.

2.1 Deep Learning

Before discussing *Bayesian* deep learning, we first briefly discuss the basics of deep learning itself. A deep neural network is a model consisting of multiple recursive functions called "layers", each layer is a function of the form $f : \mathbb{R}^p \to \mathbb{R}^q$, with p the input dimension and q the output dimension. A layer transforms the input in some way, for example a linear layer computes an affine transformation of the input using a set of learnable parameters. In this case, m is also called the number of hidden units or neurons. By alternating parameterised layers and activation layers, which compute an element-wise non-linear function ($p = q$), neural networks can be used to express complex functions. The simplest neural network is the fully connected model (also known as a feed-forward model or multi layer perceptron). It consists of multiple blocks of a linear layer followed by an activation layer:

$$f_i(\boldsymbol{x}_i) = \sigma(\boldsymbol{W}_i \boldsymbol{x}_i + \boldsymbol{b}_i), \tag{2.1}$$

with $\boldsymbol{x}_i$ the vector-valued input at block i, σ an activation function, $\boldsymbol{W}_i$ the learnable affine transformation matrix of block i, $\boldsymbol{b}_i$ a bias term. There are many activation functions, but a common choice is the rectified linear unit, which is defined as $\max(0, \boldsymbol{x}_{ij})$ computed element-wise (Nair et al., 2010).

Generally, neural networks are learned using stochastic gradient descent (SGD) or more advanced alternatives such as Adam (Kingma et al., 2014a). SGD works by first sampling a minibatch of data from the training data. Then the model is evaluated on this data by recursively computing each layer output. The intermediate layer outputs are also called features. The final output of the model is then used in combination with a label to compute an objective function (also called loss function). A common example in classification problems is the cross-entropy function. The gradients with respect to all the learnable parameters can be obtained using backpropagation (Rumelhart et al., 1986). By taking a small step in the direction of the gradient that minimises the loss, the model can be fit to a dataset.

Recent advances in deep learning have focused on learning significantly deeper and wider models. In particular, residual connections (He et al., 2016) and

normalisation layers are now fundamental components of most model structures (or "architectures"). In a residual layer, the model computes additive terms instead of freely transforming the input:

$$f_i(\boldsymbol{x}_i) = \boldsymbol{x}_i + \sigma(\boldsymbol{W}_i\boldsymbol{x}_i + \boldsymbol{b}_i). \tag{2.2}$$

Normalisation layers work by standardising an intermediate value, and they are frequently placed after a linear or residual layer. The most common one is Batch Normalisation (Ioffe et al., 2015), which normalises its input using the batch mean and standard deviation. At test time it uses a running mean and standard deviation obtained from the training set. These innovations have made it possible to train significantly deeper models, and take larger gradient descent steps which speeds up training. For example, the ResNet (He et al., 2016) demonstrated that with residual connections it is possible to scale from tens of layers to hundreds, with deeper models being at least as good as their shallower counterpart. Deep neural networks are a cornerstone of this thesis and used throughout all chapters.

2.2 Bayesian Deep Learning

Bayesian deep learning (BDL) is the name for a broad set of deep learning methods that include a Bayesian component. The archetypical model of BDL is the Bayesian neural network (BNN) (MacKay, 1992; Neal, 1996), which is a Bayesian extension of the standard neural network. This extension involves changing the parameters from point estimates to a distribution, placing a prior on the parameter distribution and updating them using Bayes rule. Predictions in a BNN are made by integrating over all possible parameter configurations. Let $p(\boldsymbol{w}|\mathcal{D}_{\text{train}})$ be the parameter distribution, $\mathcal{D}_{\text{train}} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ the dataset, $\boldsymbol{x}$ an input and y a prediction for that input:

$$p(y|\boldsymbol{x}, \mathcal{D}_{\text{train}}) = \int p(y|\boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{w}|\mathcal{D}_{\text{train}}) d\boldsymbol{w}. \tag{2.3}$$

Making exact predictions (or inferences) with a BNN is intractable in general, because there is no closed form for this integral. The integral is also hard to approximate well, for example using Monte Carlo methods, due to the large number

of parameters. One promising technique for approximating this integral is by restricting the parameter distribution to a computationally convenient form.

In this section, we start by giving an illustrative example of an approximation for learning a BNN called mean-field variational inference (VI). While we do not use the method in this thesis directly, because it does not perform well empirically, the example introduces many concepts that we depend on later. For example we also use VI to learn the model in Chapter 4. Subsequently, we consider two alternative BDL methods that are frequently used as baselines: MC dropout (Gal et al., 2016) and Deep Ensembles (Lakshminarayanan et al., 2017). In this thesis, we use MC dropout to approximate inference in Chapter 5 and compare frequently to Deep Ensembles in Chapters 3 and 4. Lastly, we look at a technique called Deep Kernel Learning (DKL) (Hinton et al., 2008; Calandra et al., 2016; Wilson et al., 2016a), where a Gaussian process is defined over a deep model's output. DKL combines a Bayesian output layer with a deterministic deep model and in Chapter 4 we investigate using it to estimate uncertainty.

2.2.1 Variational Inference

A rigorous and well performing method to approximate learning in a BNN is variational inference, where an approximating parameter distribution q is chosen that has a form that allows easy computation of the posterior predictive distribution. The distribution q is governed by learnable hyperparameters θ. We attempt to find the optimal parameters θ' that minimise the Kullback-Leibler (KL) divergence (Kullback et al., 1951) between the approximate distribution q and the true posterior distribution p:

$$\theta' = \arg\min_{\theta} \mathrm{KL}[q(\boldsymbol{w}|\theta)||p(\boldsymbol{w}|\boldsymbol{X})]. \tag{2.4}$$

In effect, VI turns an integration procedure (marginalisation) into an optimisation procedure, which is more amenable to deep learning.

We now describe an approximate inference method for BNNs known as Bayes by Backprop (BBB) (Blundell et al., 2015). In BBB, the posterior distribution is

restricted to have a diagonal covariance. This is known as the mean-field approximation (Saul et al., 1996) and leads to a significant reduction in computation as the inverse of the covariance matrix is straightforward to compute when it is diagonal. The parameter distribution $q(\boldsymbol{w}|\theta)$ can be described using two hyperparameters per model parameter: a mean μ and a standard deviation σ, such that $\theta = (\mu, \sigma)$. The method makes use of the reparametrisation trick (Kingma et al., 2013) which uses the differentiable sampling procedure:

$$\boldsymbol{w} = \mu + \sigma\epsilon, \epsilon \sim \mathcal{N}(0, \mathcal{I}). \tag{2.5}$$

From Equation 2.4, we can derive the objective function called the ELBO:

$$\begin{aligned}
\theta' &= \arg\min_{\theta} \mathrm{KL}[q(\boldsymbol{w}|\theta)||p(\boldsymbol{w}|\boldsymbol{X})] && (2.6)\\
&= \arg\min_{\theta} \int q(\boldsymbol{w}|\theta) \log \frac{q(\boldsymbol{w}|\theta)}{p(w)p(\boldsymbol{X}|\boldsymbol{w})} d\boldsymbol{w} && (2.7)\\
&= \arg\min_{\theta} \mathrm{KL}[q(\boldsymbol{w}|\theta)||p(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log P(\boldsymbol{X}|\boldsymbol{w})] && (2.8)\\
&= \arg\min_{\theta} \mathrm{ELBO}(\boldsymbol{X}, \theta). && (2.9)
\end{aligned}$$

The first term of the ELBO is the KL divergence between a distribution over the model weights, given the learned hyperparameters θ, and the prior $p(\boldsymbol{w})$ which in the case of BBB is chosen to be a Gaussian scale mixture, but also commonly set to $\mathcal{N}(0, \mathcal{I})$. The second term is called the likelihood term, and measures how well the model fits the data. With the reparametrisation trick, it is possible to compute a gradient of the ELBO with respect to θ and learn the model using SGD. Special care needs to be taken to appropriately re-weight the KL divergence term for each minibatch. The simplest method is to scale the KL term by the size of the minibatch divided by the total training set size, but several alternatives have been proposed that can lead to more stable training, such as a warm-up phase (Higgins et al., 2017).

While VI makes it possible to learn and evaluate BNNs, it introduces new hyperparameters that need manual tuning and reduces accuracy in many cases. MC Dropout (Gal et al., 2016) and Deep Ensembles (Lakshminarayanan et al., 2017) are two alternative methods for obtaining uncertainty in deep learning. These methods are simpler to implement than VI, obtain high accuracy, and are (surprisingly) competitive at estimating uncertainty.

2.2.2 MC Dropout

MC Dropout builds on the concept of dropout layers (Srivastava et al., 2014). A dropout layer sets some rows of its matrix-valued inputs to zero with probability p. More formally, it computes the Hadamard product between its input and a mask, where each row of the mask is sampled from a Bernoulli with probability p. At test time, a dropout layer computes the expected value of the Bernoulli by keeping all inputs, but multiplying them by $1 - p$. In the MC Dropout method, the dropout probability is kept at p at test time and several outputs are computed with different dropout masks. Gal et al. (2016) showed that, under mild assumptions, these outputs can be treated as samples from an approximation to the Bayesian posterior, which makes MC dropout a form of approximate inference. Training a model with dropout layers is similar in cost to training a standard model, but evaluating the model requires one forward pass per sample which can become a significant computational burden. By using the same dropout mask for several inputs, it is possible to compute samples of a joint posterior. We harness this ability for batch active learning in Chapter 5.

Uncertainty in MC dropout for classification problems is measured as the entropy of the average predictive distribution:

$$\hat{p}(y|\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} p_{\theta_i}(y|\boldsymbol{x})$$
$$H(\hat{p}(y|\boldsymbol{x})) = -\sum_{i=0}^{C} \hat{p}(y_i|\boldsymbol{x}) \log \hat{p}(y_i|\boldsymbol{x}),$$

with θ_i the dropout mask for sample i, or parameters for ensemble element i as we see in the next section.

2.2.3 Deep Ensembles

In Deep Ensembles, several models are trained with different weight initialisations and ordering of the training data. These models can be obtained by simply running a training script with several random seeds. At test time, a prediction is made by ensembling the trained models and uncertainty is quantified by looking at their

disagreement, e.g., by looking at the entropy of the average prediction or their variance. Deep Ensembles has been shown to be a very effective method in practice (Ovadia et al., 2019), and the ensemble elements show a large amount of output diversity (Fort et al., 2019). Training several models incurs additional computational cost: Deep Ensembles' memory and compute use scales linearly with the number of ensemble elements at both train and test time, although it is possible to run the ensemble elements in parallel. A significant amount of work has been done to verify to what extent Deep Ensembles are an approximate Bayesian model and if it is possible to make them more Bayesian (He et al., 2020; D'Angelo et al., 2021). In summary, there is evidence that Deep Ensembles provide a closer approximation to the Bayesian posterior predictive distribution than most standard approximate inference procedures (Wilson et al., 2020; Izmailov et al., 2021; Wilson et al., 2022). We compare to Deep Ensembles and analyse its shortcomings in Chapters 3 and 4.

2.2.4 Deep Kernel Learning

Deep kernel learning (Hinton et al., 2008; Calandra et al., 2016; Wilson et al., 2016a) is a method that combines a neural network (NN) with a Gaussian process (GP). In DKL, the data is first transformed into a feature space using a NN, then a GP is placed on this feature space. The idea is that the NN transforms the input data in such a way that it is more amenable to the GP kernel.

A commonly used kernel in GPs is the Radial Basis Function (RBF) kernel, applied on two vector-valued inputs:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = s \exp\left[-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2}{l}\right], \tag{2.10}$$

with s and l the output scale and length scale hyperparameters. When directly applied to images, and many other high dimensional inputs, this kernel is not a good measurement of the similarity of its inputs (Theis et al., 2016). However, NNs excel at transforming high dimensional data into an informative lower dimensional representation, and they are able to learn a feature space where the RBF kernel *is* a good measurement of similarity.

We now describe a version of DKL that works on large datasets and updates all its learnable parameters at each step of gradient descent ("end-to-end" learning). The model closest to what we discuss is the GPDNN (Bradshaw et al., 2017), and we make use of the inducing point variational approximation first introduced in Titsias (2009), in particular the variant discussed in Hensman et al. (2015).

Let $\mathcal{D}_{\text{train}} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ be our dataset, with $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ and $\boldsymbol{Y} \in \mathbb{R}^{N \times T}$ be a dataset of N points with input dimensionality D and output dimensionality T. In addition to regression, we also consider classification tasks. For classification tasks, T is the number of classes, with a single instance $\boldsymbol{y} \in \boldsymbol{Y}$ being a T dimensional vector of class probabilities. Let $\boldsymbol{F} \in \mathbb{R}^{N \times T}$ be the value of a T dimensional latent function at each input. For regression tasks F is the values of the underlying noiseless function the GP is modelling. The model is formed of an independent GP for each output dimension. The joint distribution over $\boldsymbol{Y}$ and $\boldsymbol{F}$, evaluated at inputs $\boldsymbol{X}$, is

$$p(\boldsymbol{Y}, \boldsymbol{F}; \boldsymbol{X}) = p(\boldsymbol{Y} \mid \boldsymbol{F}) \prod_{c=1}^{C} p(\boldsymbol{F}_{[:,c]}; \boldsymbol{X}) \tag{2.11}$$

$$p(\boldsymbol{F}_{[:,t]}; \boldsymbol{X}) = \mathcal{GP}(\mu_t(\boldsymbol{X}), k_{l_t,\theta}(\boldsymbol{X}, \boldsymbol{X})). \tag{2.12}$$

Here $\boldsymbol{F}_{[:,t]}$ refers to the t-th column vector of the matrix $\boldsymbol{F}$. $\mu_t(\cdot)$ is a mean function for output dimension t. For both regression and classification we use a constant mean function $\mu_c(\boldsymbol{X}) = \mu_t$, where μ_t is a hyperparameter. $k_{l,\theta}(\cdot, \cdot)$ is a deep kernel function with feature extractor parameters θ and base kernel $\bar{k}_{l_c}(\cdot, \cdot)$ with hyperparameters l_t specific to each output dimension (but shared for each input dimension). $p(\boldsymbol{Y} \mid \boldsymbol{F})$ is the likelihood function. For regression tasks this is defined $p(\boldsymbol{Y} \mid \boldsymbol{F}) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{Y}_{[i,:]} \mid \boldsymbol{F}_{[i,:]}, \sigma^2 \boldsymbol{I}_T)$, where σ^2 is a variance hyperparameter and $\boldsymbol{I}$ is the identity matrix. For classification tasks,

$$p(\boldsymbol{Y} \mid \boldsymbol{F}) = \prod_{i=1}^{N} p(\boldsymbol{Y}_{[i,:]} \mid \boldsymbol{F}_{[i,:]}) \tag{2.13}$$

$$p(\boldsymbol{Y}_{[i,:]} \mid \boldsymbol{F}_{[i,:]}) = \text{softmax}(\boldsymbol{F}_{[i,:]}). \tag{2.14}$$

Note that while μ_t, σ^2, and l_t are described as hyperparameters, they do not need to be specified manually but instead can be learned alongside the other model parameters.

Sparse GP Approximation and Variational Inference Exact inference for the classification likelihood is not tractable because the softmax function is not a conjugate likelihood to the GP prior. Additionally, while exact inference is possible for the regression case, the computational complexity scales cubically (due to inversion of the Gram matrix) with the number of data points, thus it is not suitable for large datasets. As an approximation, it is possible to use a sparse GP and variational inference for both regression and classification.

We use the sparse GP approximation of Titsias (2009) which augments the model with M inducing inputs, $\boldsymbol{Z} \in \mathbb{R}^{M \times J}$, where J is the dimensionality of the feature space. The associated inducing variables, $\boldsymbol{U} \in \mathbb{R}^{M \times T}$, give the function value at each inducing input. Together, the inducing inputs and inducing variables approximate the full dataset. To perform inference in this model we use the variational approximation introduced by Hensman et al. (2015). Here $\boldsymbol{Z}$ are treated as variational parameters. $\boldsymbol{U}$ are random variables with prior $p(\boldsymbol{U}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{U}_{[:,t]} \,|\, \mu_t(\boldsymbol{Z}), \bar{k}(\boldsymbol{Z}, \boldsymbol{Z}))$, and variational posterior $q(\boldsymbol{U}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{U}_{[:,t]} \mid \boldsymbol{m}_t, \boldsymbol{S}_t)$, where $\boldsymbol{m}_t \in \mathbb{R}^{\boldsymbol{M}}$ and $\boldsymbol{S}_t \in \mathbb{R}^{\boldsymbol{M} \times \boldsymbol{M}}$ are variational parameters and initialized at the zero vector and the identity matrix respectively. The approximate predictive posterior distribution at test points $\boldsymbol{X}^*$ is then

$$q(\boldsymbol{F}^* | \boldsymbol{Y}; \boldsymbol{X}, \boldsymbol{X}^*) = \int p(\boldsymbol{Y} \,|\, \boldsymbol{F}^*) p(\boldsymbol{F}^* | \boldsymbol{U}; \boldsymbol{X}^*, \boldsymbol{Z}) \prod_{t=1}^{T} q(\boldsymbol{U}_{[:,t]} | \boldsymbol{m}_t, \boldsymbol{S}_t) \, \mathrm{d}\boldsymbol{U} \, \mathrm{d}\boldsymbol{F}^*. \quad (2.15)$$

Here $p(\boldsymbol{F}^* \mid \boldsymbol{U}; \boldsymbol{X}^*, \boldsymbol{Z})$ is a Gaussian distribution for which we have an analytic expression, see Hensman et al. (2015) for details. Note that we deviate from Hensman et al. (2015) in that our input points $\boldsymbol{X}$ are mapped into feature space just before computing the base kernel, while inducing points are used as is (they are defined in feature space).

The variational parameters $\boldsymbol{Z}$, $\boldsymbol{m}_t$, and $\boldsymbol{S}_t$, alongside the feature extractor parameters θ and model hyperparameters μ_t, l_t, and σ^2, are all learned by maximizing a lower bound on the log marginal likelihood, known as the ELBO, $\mathcal{L}$. For the

variational approximation above, this is defined as

$$p(\boldsymbol{Y}; \boldsymbol{X}) \geq \mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_{q(\boldsymbol{F}_{[i,:]} \mid \boldsymbol{m}_{1\ldots T}, \boldsymbol{S}_{1\ldots T}; \boldsymbol{x}_i, \boldsymbol{Z})} \left[\log p(\boldsymbol{Y}_{[i,:]} \mid \boldsymbol{F}_{[i,:]})\right] - D_{\mathrm{KL}}(q(\boldsymbol{U}) || p(\boldsymbol{U})). \tag{2.16}$$

Both terms can be computed analytically when the likelihood is Gaussian and for classification (i.e., a non-Gaussian likelihood) we do MC sampling. Armed with this objective function, we can learn the model parameters, hyperparameters, and variational parameters using stochastic gradient descent. To accelerate optimisation we additionally use the whitening procedure of Matthews (2017).

For regression tasks, we directly use the function values $\boldsymbol{F}^*$ above as the predictions. We use the mean of $q(\boldsymbol{F}^* \mid \boldsymbol{Y}; \boldsymbol{X}, \boldsymbol{X}^*)$ as the prediction, and the variance as the uncertainty. For classification tasks we need the posterior over the class probabilities, $q(\boldsymbol{Y}^* \mid \boldsymbol{Y}; \boldsymbol{X}, \boldsymbol{X}^*)$, rather than the latent function values. Thus, we approximate the integral

$$\bar{\boldsymbol{Y}}^* = \int \mathrm{softmax}(\boldsymbol{F}^*) q(\boldsymbol{F}^* \mid \boldsymbol{Y}; \boldsymbol{X}, \boldsymbol{X}^*) \, \mathrm{d}\boldsymbol{F}^*, \tag{2.17}$$

using Monte Carlo samples, which are very fast to compute and do not require additional forward passes. The predicted class for input i is then the most likely class in $\bar{\boldsymbol{Y}}^*_{[i,:]}$.

2.3 Evaluating Uncertainty Estimation

In this section we discuss how to evaluate uncertainty estimation methods. We start by discussing the two main sources of uncertainty, aleatoric and epistemic. Subsequently, we look at calibration where we evaluate a model's predicted confidence with a labelled test set. We also discuss different usages of the concept "out of distribution" and clarify its meaning in the context of this thesis.

2.3.1 Aleatoric and Epistemic Uncertainty

We can distinguish two sources of uncertainty: aleatoric and epistemic uncertainty (also called data and model uncertainty). Aleatoric uncertainty comes from

ambiguity in a data point, it can not be reduced by observing more training data. An example of this type of uncertainty is a digit that looks like both a "1" and a "7", or inherent noise due to a sensor. We can further categorise aleatoric uncertainty as homoskedastic or heteroskedastic. Homoskedastic is constant across the dataset, for example due to a low quality compression technique or lens issue. Heteroskedastic is data point dependent, the example of the digits falls into this category. In most datasets, the aleatoric uncertainty comes from a combination of both homo- and heteroskedastic noise, but some methods can only model one or the other. Epistemic uncertainty is uncertainty that comes from a small dataset or limited model capacity. We expect this uncertainty to reduce as the model is trained on more data or increased in capacity.

For many applications, it is beneficial when a model can distinguish between epistemic and aleatoric uncertainty. For example in active learning, we are interested in labelling data points with high epistemic uncertainty, but not those with high aleatoric uncertainty. As data points with high aleatoric uncertainty are inherently ambiguous or noisy and do not belong to any class, it is not possible to label them accurately (the true label distribution is noisy). However, it is not always straightforward to assign a cause of uncertainty to either of these sources. For example, in the case of the ambiguous digit, the writer (likely) had the intent to write one or the other and including the digits author as input feature would resolve this uncertainty. Or perhaps in the context of a larger set of samples of their handwriting it would be obvious which one it is. In general, a model cannot quantify aleatoric uncertainty until it has a robust understanding of the dataset, i.e., low epistemic uncertainty. We dig deeper into this in Section 3.4.

2.3.2 Calibration

Calibration measures how well a model's predicted confidence reflects its actual accuracy, for example if the model predicts 100 times with 70% confidence, then we roughly expect 70 predictions to be correct and the remaining 30 to be incorrect.

This is desideratum 1 in Section 1.1. Calibration quality can be quantified using the Expected Calibration Error (ECE):

$$\sum_{m=1}^{M} \frac{|B_m|}{N} \left|\text{acc}(B_m) - \text{conf}(B_m)\right|, \tag{2.18}$$

with M the number of bins, N the total number of data points, B_m the data points in bin m, $\text{acc}(B_m)$ represents the average accuracy of predictions on points in bin m and $\text{conf}(B_m)$ the average confidence. The binning can be done in two ways: equal width or equal mass. In equal width binning, a set of equidistant points is chosen between 0 and 1 which form the bin boundaries, points are assigned to a bin based on their confidence. In equal mass binning, the boundaries are chosen such that each bin contains an equal number of points. Note that calibration is independent of accuracy, as a classifier that always predicts all classes with uniform probability is perfectly calibrated on a balanced dataset.

There are several important aspects of calibration to be aware of. First, the concept of calibration defined previously can only be computed in the classification setting. Second, to evaluate calibration we need to compute a notion of accuracy, however this is difficult when the class distribution of the data at test time does not overlap with the training data. As we are also interested in evaluating our model in this setting, we need an additional evaluation metric for uncertainty estimation. We discuss how to evaluate this setting in the next section. Third, it turns out that in practice it is straightforward to improve the calibration of many trained classifiers by simply temperature scaling its softmax output (Guo et al., 2017; Minderer et al., 2021). This procedure requires a small amount of additional training data, which is often available.

Another way of evaluating calibration is using selective prediction, also known as rejection classification. In selective prediction, accuracy is evaluated only on data points with a confidence above a certain threshold. In Galil et al. (2022), it was shown that the area under the receiver operating characteristic curve (AUROC) of the binary event of correctly versus incorrectly classified as function of confidence, is the most consistent way to compare models in a variety of settings. The ROC

is defined as plotting the true positive rate as a function of the false positive rate. This evaluation method can be preferable compared to ECE, because it is insensitive to post-hoc scaling, which makes it easier to compute. We use this evaluation method in Chapters 3 and 4.

2.3.3 Out of Distribution

In this subsection we look at evaluating desideratum 2 of Section 1.1: uncertainty should be high on data points that cannot be classified. The first step is to learn a model on a particular dataset. Subsequently, we use the model's uncertainty estimation as a metric for distinguishing between the test set belonging to the learned dataset, and an additional dataset that is semantically different. The expectation is that the model has high epistemic uncertainty on all points in the additional dataset and low epistemic uncertainty on points of the training set. There are several pairs of viable datasets which vary in difficulty and what they test (Nalisnick et al., 2019a). This evaluation is called "out of distribution (OOD) detection" and there are several important nuances that need to be taken into account when performing it. First, the other dataset can not be used during training. Second, the other dataset can also not be used for model selection and hyperparameter optimisation. Last, the other dataset is preprocessed using the same mechanism as the original dataset and each point should be treated independently, so approaches using typicality (Nalisnick et al., 2019c) or other multi-sample or batch level tests are not possible.

This evaluation is similar to the concept of anomaly detection, but the main difference is that anomalies have low likelihood under the data generating distribution and OOD data could have zero likelihood (i.e., outside its support). It is also similar to open set recognition (OSR) (Scheirer et al., 2012), a subfield of computer vision where it is explicitly assumed that some classes are unknown at training time. An important use case of OSR is in semantic segmentation, where each pixel is assigned to a class, but many pixels simply belong to a catch-all "background" class. The main difference between OOD detection as used in this thesis and OSR, is that we are interested in evaluating and improving uncertainty

estimation and are therefore restricted in the types of strategies that can be used to improve performance on this task.

The "out of distribution" naming of this benchmark is unfortunately vague and poorly defined. Strictly speaking, it means all data that are not *in* distribution, or more formally all data with likelihood zero under the data generating distribution. However this is a very strict definition and in the real world OOD points are usually sampled from a different, but related distribution to the training data. For example, a related distribution could be a mild domain or covariate shift. Another view is that if an image had been rejected by a labeller for this dataset, e.g., because it is too noisy or does not contain any of the classes of interest, then it can be considered OOD. However, it has also been used in the context of distribution shift, a concept we discuss in the next section. In this thesis, we consider OOD detection to be defined as distinguishing between the test sets of two semantically different datasets using a score based on uncertainty. We use this evaluation method in Chapters 3 and 4. A deeper dive into the nuances of this benchmark and potential alternatives is discussed in Section 6.1.

2.3.4 Robustness

A significant body of research uses the terms OOD robustness and OOD generalisation, see Hendrycks et al. (2021) for an overview. This is seemingly at odds with the definition of OOD in the previous section, because it does not seem possible to be able to generalise to points that are not classifiable with the current model. However, this inconsistency can be explained with more context. Instead of defining OOD as data points with zero likelihood, it can be defined as a *shift* of the training dataset. A *covariate* shift would be adding new species of dogs to the dog class, or drawings of objects instead of only photographs. When "OOD" implies a covariate shift, it is perfectly reasonable to discuss robustness and generalisation, and it is generally preferable that models retain high accuracy even under shifts. See Moreno-Torres et al. (2012), for an in-depth discussion

and examples of many types of shifts, including a discussion on which shifts are reasonable to generalise to and which are not.

Just like in OOD detection, it is important that in any evaluation in the presence of shifted data points no access is given to the exact type of shift at training time. In practice, it can be difficult to predict what changes will manifest during use of the model. This was exemplified in Finlayson et al. (2021) who show the detrimental effect that the new disease COVID-19 had on the accuracy of a sepsis-alerting model. However, access to a shift during training can significantly increase performance (Band et al., 2021) leading to evaluations that seem more positive than what can be expected in practice.

2.3.5 Parametric and Non-Parametric Models

To understand the necessity of the fourth optional desideratum, "uncertainty should remain high on parts of the input domain that we cannot generalise to based on the training data", we consider what happens in the context of uncertainty estimation in the limit of data, that is an infinitely large dataset. Aleatoric uncertainty remains constant in the limit of data, since it is irreducible. For epistemic uncertainty, we must distinguish between parametric and non-parametric models. Parametric models are models where a finite amount of parameters is fit to the data. Bayesian Neural networks fall in this category and, when trained to maximise the evidence lower bound, will have zero uncertainty everywhere except on ambiguous data points. The model will be certain even on data points that are unlike the training data. In 2D this can be visualised as data points "far away" from the rest of the dataset. This is an example of an overfit model in the context of uncertainty estimation. In non-parametric models, such as exact Gaussian processes, this does not happen and the model remains uncertain far away from the training data even in the limit of data. Note that non-parametricity is not the only way to achieve this desideratum and we look at how different models behave in the limit of data setting in Chapter 4.

2.4 Active Learning

In active learning (AL) (Cohn et al., 1996; Settles, 2009), we are given a set of data points $\mathcal{D}_{\text{pool}}$ without their corresponding labels, some resources to obtain new labels, and we want to learn the best performing model. Obtaining a label incurs a cost, for example an X-ray image that has to be diagnosed by a radiologist, and we want to be able to use our resources to obtain new labels in the most efficient way possible. Alternatively put, we want to obtain labels only for the most informative data points. We look at active learning at several moments in this thesis: *batch* active learning in Chapter 5, active learning for causal inference in Section 4.5, and active learning with large pre-trained models in Section 5.6.

AL is an iterative process: first the model is trained on the currently available dataset, then its performance is evaluated, and lastly new data is selected and labelled if performance is not deemed sufficient. Selecting new points is done by evaluating an acquisition function on the current set of unlabelled data points and learned model parameters θ:

$$\{x_1, ..., x_b\} = a(\mathcal{D}_{\text{pool}}, \theta). \tag{2.19}$$

This definition of the acquisition function assumes acquiring b points at a time. The special case of $b = 1$ is the traditional AL setup, while $b > 1$ corresponds to *batch* AL. In AL, we train our large model from scratch after each acquisition step, which can be very computationally expensive for larger models. This is necessary because just continuing with the previous set of parameters leads to poor performance, and resolving this is an open problem (Ash et al., 2020). In deep learning, batch AL is particularly compelling because it reduces expensive re-training of large neural networks.

2.4.1 Acquisition Functions

The simplest acquisition function is random sampling, where a random subset of $\mathcal{D}_{\text{pool}}$ is selected at each acquisition step. This simple function is a surprisingly strong baseline, and comes with a guarantee that the labelled subset is an unbiased

sample of the full dataset. For more interesting acquisition function, we focus on several that use some notion of uncertainty. Given a function that measures uncertainty for a single data point, we can extend it to the batch AL setting by obtaining the top b highest scoring data points:

$$a(\mathcal{D}_{\text{pool}}, \theta) = \underset{\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_b\}\in\mathcal{D}_{\text{pool}}}{\arg\max} \sum_{\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k\}} s(\boldsymbol{x}_i, \theta), \tag{2.20}$$

with s the function that measures informativeness on a single data point.

We now define the score function of the main uncertainty-based acquisition functions for classification models with a C dimensional categorical output.

1. Predictive entropy:

$$s(\boldsymbol{x}, \theta) = \sum_{c=1}^{C} p(y_c|\boldsymbol{x}) \log p(y_c|\boldsymbol{x}). \tag{2.21}$$

2. Maximum softmax probability (Hendrycks et al., 2017):

$$s(\boldsymbol{x}, \theta) = \max_c p(y_c|\boldsymbol{x}). \tag{2.22}$$

3. Maximum margin (Scheffer et al., 2001):

$$s(\boldsymbol{x}, \theta) = p(y_1|\boldsymbol{x}) - p(y_2|\boldsymbol{x}), \tag{2.23}$$

   where y_1 and y_2 are the first and second most probable predicted class.

4. BALD (Houlsby et al., 2011), expectations are over the parameter distribution:

$$s(x, \theta) = \sum_{c=1}^{C} \mathbb{E}[p(y_c|\boldsymbol{x}, \theta)] \log \mathbb{E}[p(y_c|\boldsymbol{x}, \theta)] - \mathbb{E}[\sum_{c=1}^{C} p(y_c|\boldsymbol{x}, \theta) \log p(y_c|\boldsymbol{x}, \theta)]. \tag{2.24}$$

Functions 1 to 3 can be evaluated for a deterministic model while BALD requires a probabilistic model. When BALD cannot be computed in closed-form, it is common to evaluate it using a Monte Carlo approximation by sampling from the distribution over weights (Gal et al., 2017).

A significant downside of summing individual data point scores in a batch (Equation 2.20) is that it does not consider correlations between data points. When

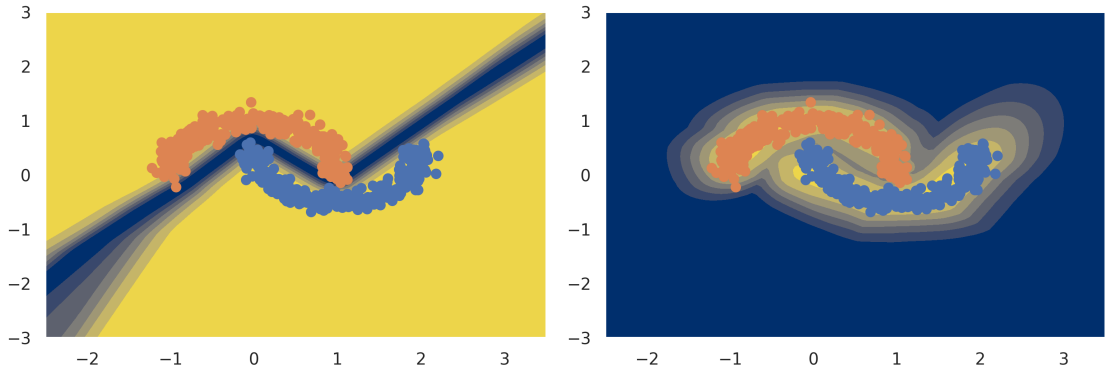
the unlabelled dataset $\mathcal{D}_{\text{pool}}$ contains many (near) duplicate elements, then the acquisition function is at risk of acquiring up to b near duplicates at each iteration as these points will have very similar acquisition scores. This problem is related to the additional desideratum 3 on computing joint uncertainties. We discuss this problem in more depth and provide an extension of BALD which resolves this problem in Chapter 5.

3

Deterministic Uncertainty Quantification

Contents

In this chapter, we introduce a deep model that is able to estimate uncertainty in a single forward pass. We call our model DUQ, Deterministic Uncertainty Quantification, and we construct it by re-examining ideas originally suggested in the 90s. By combining these with recent advances and making a number of improvements, we are able to use them in training of modern deep learning architectures. We evaluate our model against the strong and simple baseline for estimating uncertainty in Deep Learning, Deep Ensembles (Lakshminarayanan et al., 2017), and show that DUQ compares favourably on a number of evaluations, such



(a) Deep Ensembles **(b)** DUQ

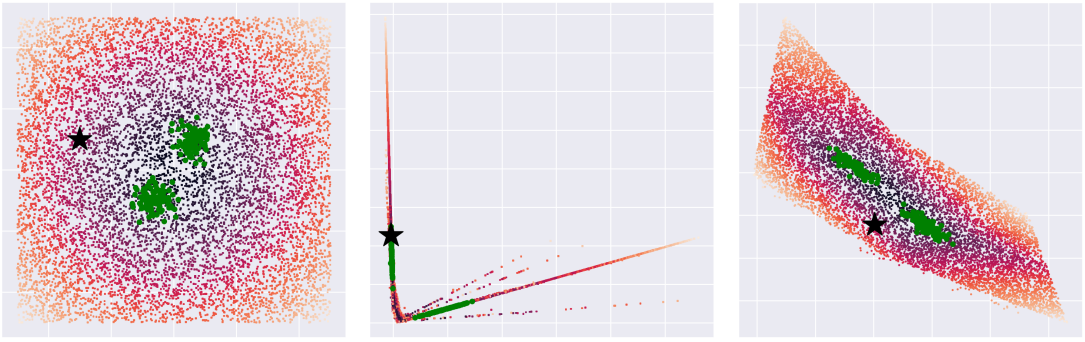
Figure 3.1: Uncertainty results on two moons dataset. Yellow indicates high certainty, while blue indicates uncertainty. DUQ is certain only on the data distribution, and uncertain away from it: the ideal result. Deep Ensembles is uncertain only along the decision boundary, and certain elsewhere.

as out of distribution (OOD) detection of FashionMNIST vs MNIST, and CIFAR vs. SVHN. In Figure 3.1, we visualise how DUQ performs on the two moons dataset. We see that DUQ is only certain on the training data, and its certainty decreases away from it. Deep Ensembles are not able to obtain meaningful uncertainty on this dataset, because of a lack of diversity in the different models in the ensemble. The code for this chapter is publicly available.[1]

DUQ consists of a deep model and a set of feature vectors corresponding to the different classes (*centroids*). A prediction is made by computing a kernel function, a distance function, between the feature vector computed by the model and the centroids. This type of model is called an *RBF network* (LeCun et al., 1998a) and uncertainty is measured as the distance between the model output and the closest centroid. A data point for which the feature vector is far away from all centroids does not belong to any class and can be considered out of distribution (following the definition of Section 2.3.3).

The model is trained by minimising the distance to the correct centroid, while maximising it with respect to the others. This incentivises the model to put the features of the training data close to a particular centroid, however there is no mechanism that dictates what should happen away from the training data. We still

[1] `https://github.com/y0ast/deterministic-uncertainty-quantification`



(a) Input **(b)** Trained without constraint **(c)** Trained with constraint

Figure 3.2: A 2D classification task where the classes are two Gaussian blobs (drawn in green), and a grid of unrelated points (coloured according to their log-probability under the data generating distribution). We additionally mark a specific point with a star. In (b), the features as computed by an unconstrained model. In (c), the features computed by a model with residual connections and spectral normalisation. The objective for the unconstrained model introduces a large amount of distortion of the space, collapsing the input to a single line, making it almost impossible to use distance-sensitive measures on these features. In particular, the star moves from an unrelated area in input space on top of class data in feature space. In contrast, the constrained mapping maintains the relative distances of the other points.

need to enforce that DUQ is sensitive to changes in the input, such that we can reliably detect out of distribution data and avoid mapping out of distribution data to in distribution feature representations — an effect we call *feature collapse*. Feature collapse is exemplified in Figure 3.2 and is defined as the feature representations of two data points being very close (for example as measured by the mean squared distance), despite the two points not sharing a similar label. This tends to happen for out of distribution data that the model has not seen during training: out of distribution data is mapped on top of in distribution data in feature space (see Figure 3.2). DUQ avoids feature collapse by regularising the Jacobian with respect to the input, as was first introduced by Drucker et al. (1992). Note that sensitivity can be at odds with obtaining a high accuracy, an effect we discuss in Section 3.1.4.

In practice, RBF networks prove difficult to optimise, because of instability of the centroids and a saturating loss. We propose to make training stable by updating the centroids using an exponential moving average of the feature vectors of the data points assigned to them, as was introduced in van den Oord et al. (2017). We use a “one vs the rest” loss function minimising the distance to the correct centroid,

while maximising the other distances. These two changes stabilise training and lead to accuracies that are similar to the common softmax and cross entropy setup on standard datasets such as FashionMNIST and CIFAR-10.

Uncertainty quantification in deep neural networks with a softmax output is generally done by measuring the entropy of the predictive distribution, so the maximally uncertain output is achieved by uniformly assigning probabilities over all the classes. In practice, the model does not assign uniform probability away from the training data, but instead extrapolates confidently (Gal, 2016; Hein et al., 2019). The only uncertainty that can reliably be captured by looking at the entropy of the softmax distribution is aleatoric uncertainty (see Section 3.4). In DUQ, it is possible to predict that none of the classes seen during training is a good fit, when the distance between the model output and all centroids is large.

The contributions of this chapter are as follows:

- We stabilise training of RBF networks and show, for the first time, that these type of models can achieve competitive accuracy versus softmax models.
- We show how two-sided Jacobian regularisation makes it possible to obtain reliable uncertainty estimates with RBF networks.
- We obtain excellent uncertainty in a single forward pass while accuracy is competitive with a softmax model.

3.1 Methods

DUQ consists of a deep feature extractor, such as a ResNet (He et al., 2016), but without the last linear and softmax layer. Instead, it has one learnable weight matrix W_c per class, c. The prediction is computed as the exponentiated distance between the model output and the centroids:

$$k_c(f_\theta(\boldsymbol{x}), \boldsymbol{e}_c) = \exp\left[-\frac{\frac{1}{n}||\boldsymbol{W}_c f_\theta(\boldsymbol{x}) - \boldsymbol{e}_c||_2^2}{2\sigma^2}\right], \tag{3.1}$$

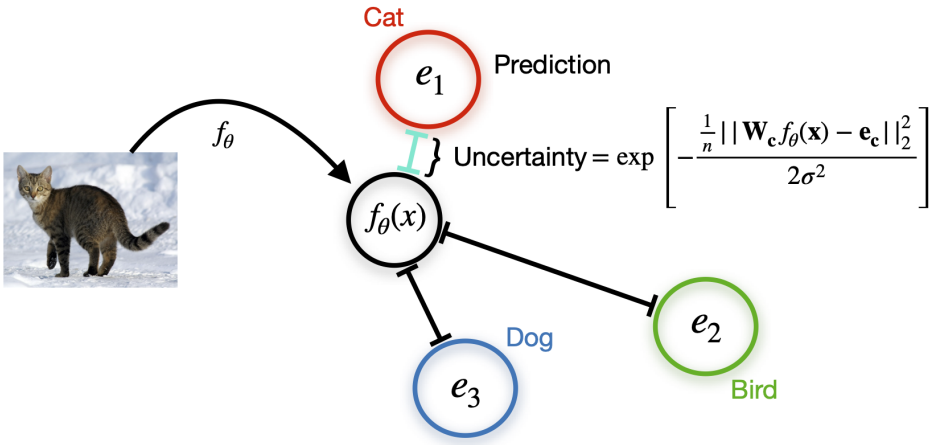


Figure 3.3: A depiction of the architecture of DUQ. The input is mapped to feature space, where it is assigned to the closest centroid. The distance to that centroid is the uncertainty.

with $f_\theta : \mathbb{R}^p \to \mathbb{R}^d$ our model, p the input dimension, d the feature dimension, and parameters θ. $\boldsymbol{e}_c$ is the centroid for class c, a vector of length n (computed following Equation 3.4). $\boldsymbol{W}_c$ is a weight matrix of size n (centroid size) by d (feature extractor output size) and σ a hyperparameter also called the length scale. This function is also referred to as a Radial Basis Function (RBF) kernel. The class dependent weight matrix improves accuracy and uncertainty empirically. A prediction is made by taking the class c with the maximum correlation (minimum distance) between data point $\boldsymbol{x}$ and class centroids $\boldsymbol{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_C\}$ (similar to Snell et al. (2017)):

$$\arg\max_c k_c(f_\theta(\boldsymbol{x}), \boldsymbol{e}_c). \tag{3.2}$$

We define uncertainty in this model as the distance to the closest centroid, i.e., replacing the arg max operator by a max in Equation 3.2.

The loss function is the sum of the binary cross entropy between each class' kernel value $K_c(\cdot, \boldsymbol{e}_c)$, and a one-hot (binary) encoding of the label. For a particular data point $\{\boldsymbol{x}, \boldsymbol{y}\}$ in our dataset $\mathcal{D}_{\text{train}} = \{\boldsymbol{X}, \boldsymbol{Y}\}$:

$$L(\boldsymbol{x}, \boldsymbol{y}) = -\sum_c y_c \log(k_c) + (1 - y_c) \log(1 - k_c) \tag{3.3}$$

where we shortened $k(f_\theta(\boldsymbol{x}), \boldsymbol{e}_c)$ as k_c, and assume $\boldsymbol{y}$ is one-hot encoded, so $y_c \in \{0, 1\}$. During training, we average the loss over a minibatch of data points, and

perform stochastic gradient descent on θ and $\mathcal{W} = \{\boldsymbol{W}_1, \cdots, \boldsymbol{W}_c\}$. The class centroids, $\boldsymbol{E}$, are updated using an exponential moving average of the feature vectors of data points belonging to that class. If the model parameters, θ and $\mathcal{W}$, are held constant, then this update rule leads to the closed form solution for the centroids that minimises the loss. With $n_{c,t}$ is the number of data points assigned to class c in minibatch t, $\boldsymbol{x}_{c,t,i}$ is element i of a minibatch at time t, with class c. γ is the momentum, which we set to 0.99 or 0.999 depending on the application: 0.99 learns faster but is less stable than 0.999. Then,

$$N_{c,t} = \gamma * N_{c,t-1} + (1-\gamma) * n_{c,t} \tag{3.4}$$

$$\boldsymbol{m}_{c,t} = \gamma * \boldsymbol{m}_{c,t-1} + (1-\gamma) \sum_i \boldsymbol{W}_c f_\theta(\boldsymbol{x}_{c,t,i}) \tag{3.5}$$

$$\boldsymbol{e}_{c,t} = \frac{\boldsymbol{m}_{c,t}}{N_{c,t}}. \tag{3.6}$$

This method of updating centroids was introduced in the Appendix of van den Oord et al. (2017). The high momentum leads to stable optimisation that is robust to initialisation.

Due to the one-vs-rest loss, the proposed setup leads to the centroids being pushed further away at each update, without converging to a stable point (further away feature representations reduce the loss). We avoid this by regularising the l_2 norm of θ. This restricts the model to sensible solutions and aids optimisation.

3.1.1 Gradient Penalty

As discussed at the start of this chapter, without further regularisation deep networks are prone to feature collapse. We find that it can be avoided by regularising the representation map using a gradient penalty. Gradient penalties were first introduced to aid generalisation in Drucker et al. (1992), who named it "double backpropagation". Recently, this type of penalty has been used successfully in training Wasserstein GANs (Gulrajani et al., 2017) to regularise the Lipschitz constant.

In our setup, we consider the following two-sided penalty:

$$\lambda \cdot \left[||\nabla_x \sum_c k_c||_2^2 - 1 \right]^2, \tag{3.7}$$

where $|| \cdot ||_2$ is the l_2 norm and the targeted Lipschitz constant is 1. The penalty is two-sided because this loss term pushes the norm up to 1 if it is below, but down to 1 if it is above.

Empirically, we found that computing the penalty on $\nabla_x \sum_c k_c$ works well. However, there are two other candidates to enforce the penalty on: $\nabla_x K$, the vector of kernel distances and $\nabla_x f_\theta(\boldsymbol{x})$, the feature vector output of the feature extractor. At first sight, these targets might actually be preferential. Sensitivity of $f_\theta(\boldsymbol{x})$ ought to be sufficient to obtain the out of distribution sensitivity properties we desire. Computing the Jacobian of a vector valued output is expensive using automatic differentiation. To evaluate the two alternative candidates we turn to the Hutchinson's Estimator (Hutchinson, 1990), which allows us to estimate the trace of the Jacobian by computing the derivative of random projections of the output. This approach was previously discussed in the context of making neural networks more robust by Hoffman et al. (2019).

While we were able to get good uncertainty on the two moons dataset using both alternative targets, the results were not consistent. We attempted to reduce the variance of the Hutchinson's estimator by using the same random projection for each element in the batch, which worked well on two moons, but lead to unsatisfactory results on larger scale datasets. In conclusion, we found that while $\nabla_x \sum_c k_c$ is not a priori the best place to compute the gradient penalty (compared to the two alternatives discussed above), it is still preferable over the noise that comes from applying Hutchinson's estimator on any of the alternatives, at least in our experiments.

The two-sided penalty was introduced by Gulrajani et al. (2017), who mention that despite a one-sided penalty being sufficient to satisfy their requirements, the two-sided penalty proved to be better in practice. The one-sided penalty, which only pushes the norm down to 1, is defined as:

$$\lambda \cdot \max(0, ||\nabla_x \sum_c k_c||_2^2 - 1). \tag{3.8}$$

In Section 3.3.1, we show the difference between the single and two-sided penalties experimentally. We find the two-sided penalty to be ideal for enforcing sensitivity, while still allowing strong generalisation.

An alternative method of enforcing sensitivity is by using an invertible feature extractor. A simple and effective method of doing so is by using invertible layers originally introduced in Dinh et al. (2014). Using these type of layers leads to strong results on the two moons dataset as seen in Figure 3.5c. Unfortunately, it is difficult to train reversible models on higher dimensional datasets. Without dimensionality reduction, such as max pooling, the memory usage of these type of networks is unreasonably high (Jacobsen et al., 2018). We found it impossible to obtain strong accuracy and uncertainty using these type of models, indicating that dimensionality reduction is an important component of why these models work.

3.1.2 Intuition about Gradient Penalty

A gradient penalty enforces smoothness, limiting how quickly the output of a function changes as the input $\boldsymbol{x}$ changes. Smoothness is important for generalisation, especially if we are using a kernel which depends on distances in the representation space. It is simple to show that regularising the l_2 norm of the Jacobian, J, enforces a Lipschitz constraint at least locally, since for a small region around $\boldsymbol{x}$ and a model g we have $g(\boldsymbol{x} + \epsilon) - g(\boldsymbol{x}) \simeq J_g(\boldsymbol{x})\epsilon \leq ||J(\boldsymbol{x})||_2||\epsilon||_2$.

However, smoothness still leaves us vulnerable to the feature collapse problem outlined earlier, where multiple inputs are mapped to the same $g(\boldsymbol{x})$. Lipschitz smooth functions can collapse their inputs — the constant function $g(\boldsymbol{x}) = c$ is Lipschitz for any Lipschitz constant L. Collapsing features can be beneficial for accuracy, but it hurts our ability to perform out of distribution detection, since it has the potential to make input points indistinguishable in the representation space. We find empirically in our work that the two-sided penalty is extremely important: using the one-sided penalty, i.e, enforcing only smoothness, is not sufficient to produce the sensitive behaviour we want in our representation. This can be seen in Figure 3.5b, in contrast to Figure 3.1b with the two-sided penalty.

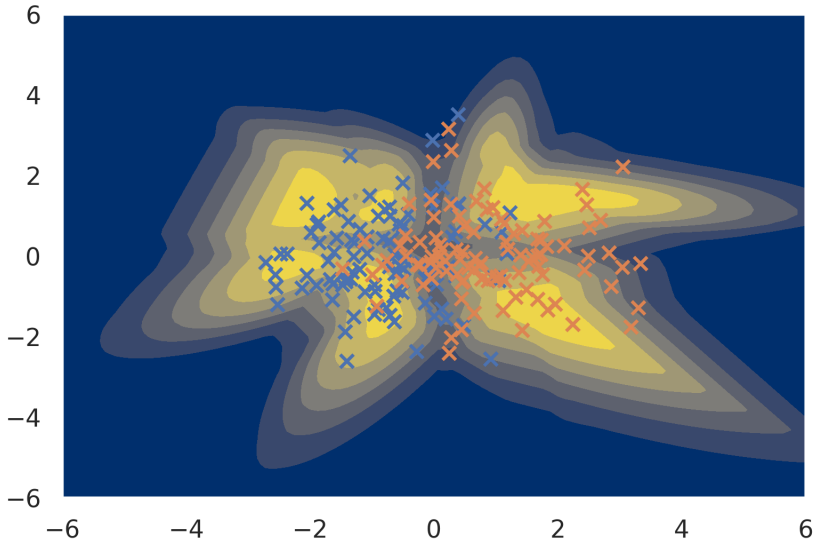


Figure 3.4: The uncertainty learned by DUQ on a simple problem of classifying samples from two overlapping Gaussian distributions. Yellow indicates certainty, while blue indicates uncertainty. There is significant aleatoric uncertainty due to the overlap between the classes. DUQ can express high aleatoric uncertainty by placing centroids close to each other in feature space, and is able to learn this in practice if the task needs it, as shown here by the higher uncertainty around the 0 mark on the x-axis.

By keeping the norm of the Jacobian above some value, intuitively we encourage sensitivity of the learnt function, by preventing it from collapsing to a locally constant function, ignoring all changes in the input space. We find empirically that the two-sided penalty is important for preserving out of distribution performance compared to a one-sided penalty.

3.1.3 Quantifying Epistemic and Aleatoric Uncertainty

DUQ can capture both aleatoric and epistemic uncertainty. Informally, when a point is far from all centroids in feature space there is epistemic uncertainty. While aleatoric uncertainty can be expressed by placing centroids so that they are close in feature space (see Figure 3.4) and mapping a data point close to both of them. This can happen when two classes are inherently difficult to distinguish. It is important that the centroids are close in feature space, because otherwise the model would not map an input in between them as it incurs a large loss, due to the exponential factor in Equation 3.3. Other forms of aleatoric uncertainty that are not inherent

class ambiguity are not quantifiable, and we also currently do not have a formal way to distinguish between these two kinds of uncertainty in DUQ. Solving this problem is an interesting direction for future research.

3.1.4 Why Sensitivity can be at odds with Classification

In this section, we analyse the trade-offs and assumptions encoded in detecting out of distribution inputs. Consider fitting a model on a problem with two features, x_1 and x_2, both sampled from a unit Gaussian, and output y, such that $y = \text{sign}(x_1) * \epsilon$, where ϵ is noise with a low probability of flipping the label. The optimal decision function in terms of the empirical risk, no matter the algorithm, is the function $f(x_1, x_2) = \text{sign}(x_1)$. But this says nothing about the out of distribution behaviour. What happens if we now see the input $x_1, x_2 = 1, 1000$? By our definition of the problem, this is out of distribution, as it lies many standard deviations away from the observed data. But should it be detected as out of distribution? The data does not define what could be given as the input, at least if we take a conventional empirical risk minimisation approach.

In this situation, it seems natural to prefer the kind of decisions which would be made by a generative model, for example. If x_1 and x_2 represent medical data, then presumably a highly abnormal value for x_2 is notable, and we would like to detect it. However, if x_2 is a truly irrelevant variable, say, the temperature on the surface of a distant planet, then presumably our model is correct to ignore its value, even if the value of the irrelevant variable is highly abnormal. When training using empirical risk minimisation, features not relevant to classification accuracy can simply be ignored by the feature extractors of a neural network. This makes out of distribution detection more difficult using feature space methods, even those that use a distance loss as we do. It is important to note that there is a potential tension here with classification accuracy. Enforcing sensitivity can make accurate classification harder because it forces the model to represent changes in input — as in the example above, these may be irrelevant to the causal structure of the problem. If we know about invariances that are appropriate for the problem at hand, we

can enforce these by corresponding construction of the network. For example, we enforce translation invariance by using convolutional networks in this chapter.

3.2 Related Work

Aside from using discriminative models, there have also been attempts at finding out of distribution data using generative models. Nalisnick et al. (2019a) showed that simply measuring the likelihood under a learned data density model does not allow detecting OOD data for some difficult pairs of datasets, such as CIFAR-10 and SVHN. Later work showed that this is particular to flow-based models (Kirichenko et al., 2020). Recently, a more advanced approach that involves separating the likelihood of the semantic foreground from the background did show promising results on selected datasets (Ren et al., 2019). While generative models are a promising avenue for out of distribution detection, they are significantly more expensive and difficult to train than classification models.

Our approach is distinct from both ensembles/Monte Carlo methods, which aim to find different explanations for the data and increase uncertainty when these disagree, and generative models which model the data distribution directly. Instead, our approach is more related to pre-deep learning kernel methods (Schölkopf et al., 2000; Quinn et al., 2014), such as Gaussian processes which revert to a prior away from data, and Support Vector Machines, where the distance to the separating hyperplane is informative of the uncertainty, at least in low dimensions. These approaches have never scaled to high dimensional data, because of a lack of well performing kernel functions.

The decision function based on kernel distances was first used in the context of convolutional neural networks by LeCun et al. (1998a). They were quickly abandoned for softmax models, because they were difficult to scale and optimise with gradient-based approaches due to saturating gradients and unstable centroids. Notable improvements in our work over the original are the updating mechanism of the centroids and the loss function that is based on a multivariate Bernoulli, solving the problems of unstable centroids and saturating gradients.

Regularising the Jacobian has a long history, starting with Drucker et al. (1992) and more recently Ross et al. (2018). Both papers aim to regularise the l_2 norm of Jacobian down to zero. In the first case to obtain better generalisation, while the second paper aims to achieve adversarial robustness and interpretability. In neither case are the authors interested in *increasing* the Jacobian. Gulrajani et al. (2017) showed how a gradient penalty can be applied to training GANs with the Wasserstein distance, which was a more scalable and simpler alternative to weight clipping. They use the double-sided penalty and mention it works better in practice. Follow-up work has analysed the penalty in more detail and concluded that, contrary to our case, for training Wasserstein GANS the one-sided penalty is preferable theoretically and practically (Petzka et al., 2017; Jolicoeur-Martineau et al., 2019).

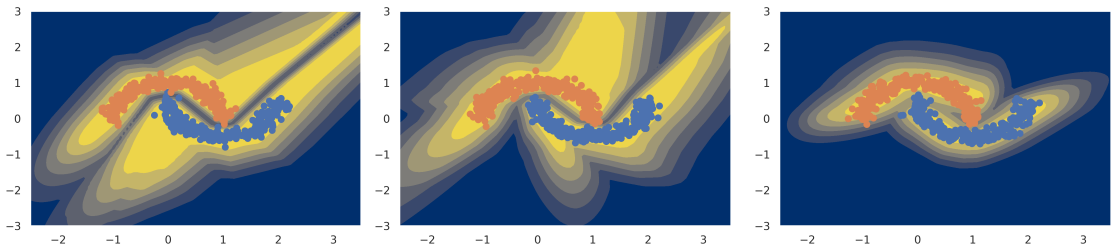
3.3 Experiments

We start by showing the behaviour of DUQ in two dimensions, with the two moons dataset and show the effect of leaving out the gradient penalty and using a one-sided penalty. We continue by looking at the out of distribution detection performance for some notable difficult dataset pairs (Nalisnick et al., 2019a), such as FashionMNIST vs MNIST, and CIFAR-10 vs SVHN. Lastly, we study sensitivity to two important hyperparameters: the length scale σ and gradient penalty weight λ and propose how to tune them without relying on example OOD data.

3.3.1 Two Moons

We use the scikit-learn (Pedregosa et al., 2011) implementation of this dataset and describe the model architecture and optimisation details in Appendix A.1.1. For colouring the visualisations, we normalise the colour map within the figure.

The result of our model trained with a two-sided gradient penalty is shown in Figure 3.1b. The uncertainty is exactly as one would expect for the two moons dataset: certain on the training data, uncertain away from it and in the heart within the two moons. The difference with Deep Ensembles, which is certain everywhere except on the decision boundary, is striking (Figure 3.1a). The uncertainty for



(a) DUQ - No penalty **(b)** DUQ - One-sided penalty **(c)** DUQ - Reversible model

Figure 3.5: Uncertainty results for two variations of DUQ: left without gradient penalty, middle with a one-sided gradient penalty ($\lambda = 1$), and right with a fully reversible model as feature extractor (Dinh et al., 2014, 2016). Yellow indicates certainty, while blue indicates uncertainty. Without gradient penalty and a one-sided gradient penalty are significantly worse than DUQ with a two-sided penalty. The reversible model works well too, but does not scale to larger datasets

DUQ is quantified as the distance to the closest centroid (*max* over the kernel distances), the uncertainty for Deep Ensembles is computed as the predictive entropy of the average output. The ensemble elements were trained separately using the same model as described in Appendix A.1.1, but without $l2$ regularisation to encourage diverse solutions.

Discussion While Figure 3.1b is an impressive result in deep learning, it is worth highlighting that Gaussian processes are able to obtain such result too. A good visualisation can be found in Bradshaw et al. (2017). Interestingly, even though Deep Ensembles have been successfully applied to many large datasets (Ovadia et al., 2019), they fail to estimate uncertainty well on the two moons dataset. Due to the simplicity and low dimensionality of this dataset, each element of the ensemble learns the exact same function and there is no (or very little) uncertainty left.

Gradient Penalty In Section 3.1.1, we discussed the two-sided gradient penalty. Figure 3.5 highlights why it is important. In Figure 3.5a, we visualise the result of having no gradient penalty, which shows that the model is certain even far away from the data. While Figure 3.5b shows that the uncertainty does not improve when only a one-sided penalty is applied. In both cases, there are 'blobs' sticking out of the training data domain that are also classified with high certainty.

Hyperparameters We found classification performance on two moons to be insensitive to our setting of the gradient penalty weight λ, likely because of the simplicity of the two moons dataset. So we simply set it to 1. For the uncertainty visualisation, we found it important to set the length scale to be small (in the interval $[0.05, 0.5]$), despite accuracy not being affected by this hyperparameter. In the following experiments, we will discuss methods for picking the length scale and the weight of the gradient penalty λ.

3.3.2 FashionMNIST vs MNIST

In this experiment, we assess the quality of our uncertainty estimation by looking at how well we can separate the test set of FashionMNIST (Xiao et al., 2017) from the test set of MNIST (LeCun et al., 1998b) by looking only at the uncertainty predicted by the model. We train our model on FashionMNIST, and we expect it to assign low uncertainty to the FashionMNIST test set, but high uncertainty to MNIST, since the model has never seen that dataset before, and it is very different from FashionMNIST.

During evaluation, we compute uncertainty scores on both test sets and measure for a range of thresholds how well the two are separated. As in previous work (Ren et al., 2019), we report the AUROC metric, where a higher value is better and 1 indicates that all FashionMNIST data points have a higher certainty than all MNIST data points. We picked FashionMNIST vs MNIST, because it is a notably difficult dataset pair (Nalisnick et al., 2019a), while MNIST vs NotMNIST (Bulatov, 2011) is comparatively simple.

Experimental Setup Our model is a three layer convolutional network, and we report all architectural and optimisation details in Appendix A.1.2. It is important to note that at test time we set Batch Normalisation to evaluation mode, meaning that we use the mean and standard deviation of the feature activations computed from the training set (i.e., FashionMNIST). It is unlikely that in practice we would get an entire batch of (uncorrelated) OOD points, so we can not normalise using

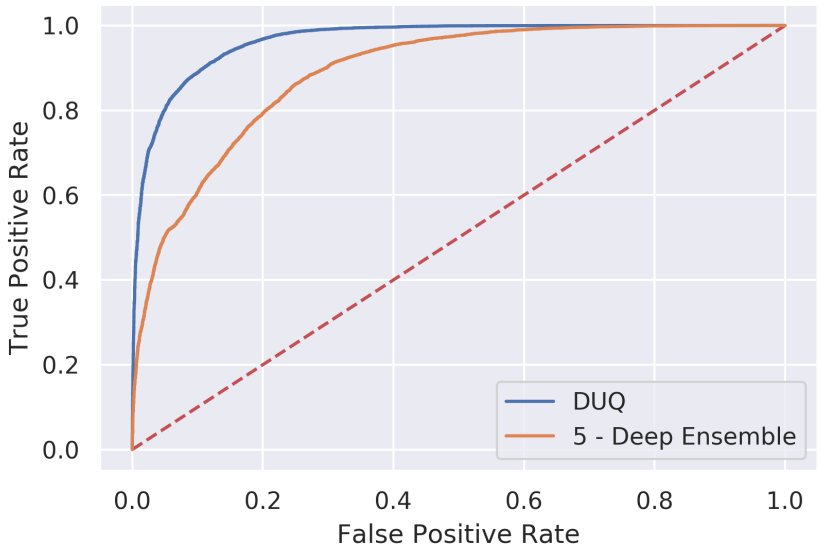


Figure 3.6: ROC curve for DUQ trained on FashionMNIST and evaluated on FashionMNIST and MNIST. The task is to separate these datasets based on uncertainty estimates. DUQ does very well here, but we see that gap closes for the harder dataset CIFAR-10 in Section 3.3.3.

test time batch statistics.[2] Further, we use the same data normalisation for the out of distribution set as the in distribution set. Skipping either of these steps makes the problem artificially simple.

Length Scale Most hyperparameters, such as the learning rate or weight decay parameter, can be set using the standard train/validation split. However, there are two hyperparameters that are particularly important: the length scale σ and the gradient penalty weight λ. We set the length scale by doing a grid search over the interval $(0, 1]$ while keeping $\lambda = 0$. We pick the value that leads to the highest validation accuracy. Following this process, we found that a length scale of 0.1 leads to the highest accuracy, as measured over five runs. While this process might not result in a length scale that leads to the best OOD performance, it works well in practice.

[2]Just one data point needs to have significantly different activation statistics for the entire batch to be easily detectable.

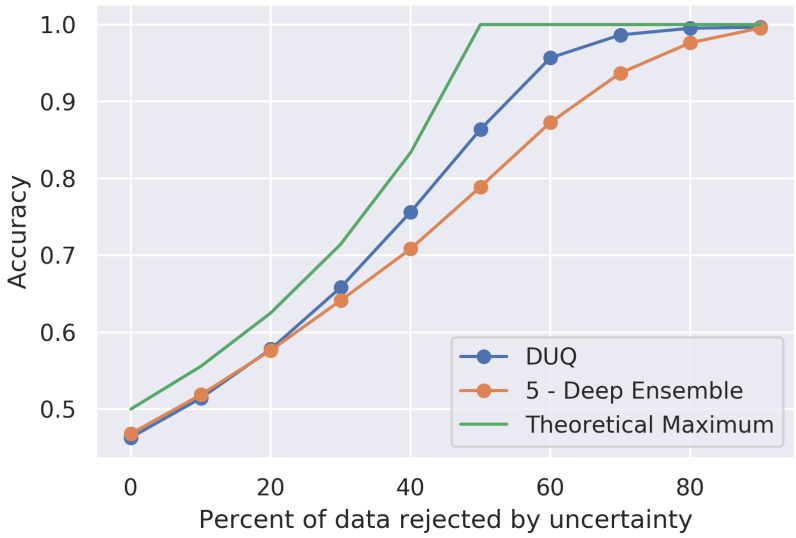


Figure 3.7: Rejection classification plot: accuracy on a combination of FashionMNIST and MNIST test sets. The x-axis indicates the proportion of data rejected based on the uncertainty score. The theoretical maximum is computed from a classifier with 100% accuracy on FashionMNIST and rejects all MNIST points first.

Gradient Penalty Setting the λ parameter is more involved: from Section 3.1.4, we know that the accuracy can suffer as a result from gaining the ability to do out of distribution detection, so we cannot rely on it to select the best λ. We also cannot use the AUROC score on the MNIST dataset, because that would give the method an unfair advantage: we cannot assume access to the OOD set in advance in practice.[3] Instead, we use a third dataset on which we evaluate the AUROC and select our λ values based on that. We follow previous work and use NotMNIST as the third dataset for this pair (Ren et al., 2019). The results can be seen in Table 3.1. As expected, the accuracy goes down as λ increases, and we also observe that the best AUROC result for NotMNIST coincides with the best score for MNIST, which shows that the strategy of selecting a hyperparameter based on the NotMNIST dataset is reasonable. We note that while NotMNIST generalises to MNIST, we cannot rely on this property in general. Therefore, we propose an alternative method for model selection based on predictive uncertainty in Section 3.3.3.

[3]If we do assume access, then we can trivially train a binary classifier on the original and OOD set.

Comparison We show our results and compare with alternative methods in Table 3.2. Our proposed method, DUQ, outperforms all other classification based methods in terms of OOD detection. The only method that is better is LL ratio (Ren et al., 2019), which is based on generative models. These type of models are more computationally costly to train than DUQ. The PixelCNN++ (Salimans et al., 2017) used by LL ratio for FashionMNIST uses 2 blocks of 5 gated ResNet layers, while our model is a simple three layer convolutional network. An alternative, competitive approach is Mahalanobis Distance (Lee et al., 2018), which computes a distance in the feature space of a pretrained softmax/cross entropy model in combination with a number of dataset specific augmentations. The method relies on hyperparameter tuning using 1,000 samples of the out of distribution dataset.

The difference in AUROC between our Deep Ensemble result and Ren et al. (2019)'s is due to using different architectures. For a fair comparison, we use the same architecture for the ensemble elements as for DUQ (replacing the class dependent final layer by the usual single linear layer). In Figure 3.6, we show the complete ROC curve for our implementation of Deep Ensembles and DUQ. We see that DUQ outperforms Deep Ensembles at all chosen rates.

Accuracy and Gradient Penalty To confirm that training using DUQ's distance based output achieves competitive accuracy, we train two models using our architecture: the standard softmax and cross entropy set up and DUQ with $\lambda = 0$. We obtained $92.4\% \pm 0.1$ accuracy for the softmax model, and for our proposed set up $92.4\% \pm 0.2$, both averaged over five runs. The results show that we can obtain competitive accuracy using DUQ, resolving previous problems with RBF networks. In Table 3.1, we show how accuracy changes for an increasingly weighted gradient penalty. The accuracy only degrades slightly, while AUROC is improved.

Rejection Classification In Figure 3.7, we visualise how well these algorithms work in a more realistic scenario. We combine the FashionMNIST and MNIST test sets, then we reject a certain portion of the combined dataset by uncertainty score. Next, we compute the accuracy on the remaining data for each portion,

λ	Acc (FM)	AUROC (NM)	AUROC (M)
0	$92.4\% \pm .2$	$0.933 \pm .009$	$0.948 \pm .004$
0.05	$\mathbf{92.4\% \pm .2}$	$\mathbf{0.946 \pm .018}$	$\mathbf{0.955 \pm .007}$
0.1	$92.4\% \pm .1$	$0.938 \pm .0018$	$0.948 \pm .005$
0.2	$92.2\% \pm .1$	$0.945 \pm .019$	$0.944 \pm .011$
0.3	$92.3\% \pm .1$	$0.944 \pm .013$	$0.941 \pm .011$
0.5	$92.0\% \pm .1$	$0.946 \pm .014$	$0.932 \pm .009$
1.0	$91.9\% \pm .1$	$0.945 \pm .018$	$0.934 \pm .006$

Table 3.1: FM stands for FashionMNIST, NM for NotMNIST, and M for MNIST. The results are mean/standard error computed from 5 experiment repetitions. We show AUROC for separating FashionMNIST from NotMNIST and MNIST; higher is better. We see that the gradient penalty improves AUROC performance slightly, but performance on this dataset pair is already very strong.

considering all predictions on the OOD MNIST set to be incorrect. We expect the accuracy to go up as we reject more of the data points on which the model is uncertain. Ideally, we reject the incorrectly classified FashionMNIST points and all MNIST points. The *Theoretical Maximum* is computed by assuming a model that has perfect accuracy on the FashionMNIST test set **and** is able to reject all MNIST data before any FashionMNIST data. This experiment combines out of distribution detection, with detecting difficult to classify data points, which is closer to actual deployment scenarios than the AUROC metric, and also a suggested practically informed evaluation method by Filos et al. (2019). Note that the ensemble model has an accuracy of 93.6% on FashionMNIST, giving it a 1.2% head start on DUQ, which has an accuracy of 92.4%. We see that DUQ outperforms Deep Ensembles in this more realistic scenario.

3.3.3 CIFAR-10 vs SVHN

In this section we look at the CIFAR-10 dataset (Krizhevsky et al., 2014), with SVHN (Netzer et al., 2011) as OOD set. We use a ResNet-18 (He et al., 2016) as feature extractor $f_\theta(\cdot)$, specifically the version provided by PyTorch (Paszke et al., 2019) with some minor modifications: we use 64 filters in the first convolutional layer, and skip the first pooling operation and last linear layer. CIFAR-10 is a difficult dataset for out of distribution detection for several reasons. There is a

Method	AUROC
DUQ	0.955
LL ratio (generative model)	0.994
Single model	0.843
5 - Deep Ensembles (ours)	0.861
5 - Deep Ensembles (ll)	0.839
Mahalanobis Distance (ll)	0.942

Table 3.2: Results on FashionMNIST, with MNIST as OOD set. Deep Ensembles is by Lakshminarayanan et al. (2017), Mahalanobis Distance by Lee et al. (2018), LL ratio by Ren et al. (2019). Results marked by (ll) are obtained from Ren et al. (2019), (ours) is implemented using our architecture. Single model is trained with softmax/cross entropy.

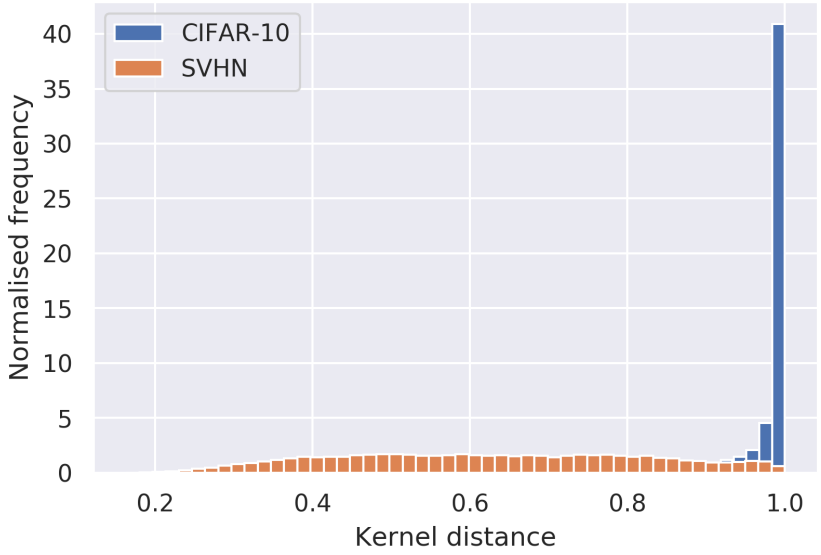


Figure 3.8: A histogram of uncertainty estimates as computed using DUQ ($\lambda = 0.5$). CIFAR-10 and SVHN are clearly separated. The counts are normalised, because the SVHN test set is significantly larger than CIFAR-10's.

significant amount of data noise: some dog and cat examples are not distinguishable using only 32 by 32 pixels. The training set is small compared to its complexity, making it easy to overfit without data augmentation.

Experimental Setup As in the previous section, we tune the length scale using the accuracy on the validation set, and find that 0.1 works best from a range of $[0.05, 1]$. We train for a fixed 75 epochs and reduce the learning rate by a factor of 0.2 at 25 and 50 epochs. We use random horizontal flips and random crops as data augmentation and find that this is enough regularisation to prevent the model from

overfitting. All architectural and optimisation details are described in Appendix A.1.3. We obtain an accuracy of $94.1\% \pm 0.2$ using the standard softmax/cross entropy loss. A Deep Ensemble of several softmax models obtains an accuracy of 95.2%. DUQ without a gradient penalty ($\lambda = 0$) obtains $94.2\% \pm 0.2$ accuracy, while accuracy of DUQ with $\lambda = 0.5$ is $93.2\% \pm 0.4$.

Gradient Penalty For CIFAR-10, we do not use a third dataset to set λ. Instead, we avoid using more data and use in distribution uncertainty. We measure this using the AUROC of detecting correctly and incorrectly classified validation set data points using the predicted uncertainty. We found that optimising λ using this procedure also transfers to λ values that lead to strong out of distribution detection performance. In general, this approach performs a bit worse than using a third dataset, but can be preferable when it is difficult to find an appropriate out of distribution dataset with the same characteristics as those encountered during deployment. Imagine a particular difficult traffic situation or an MRI scan which shows a new type of disease, these scenarios have no reasonable out of distribution set available. Generative models are not able to take this approach, because they do not have predictive uncertainty. Even if we use a hybrid model (Nalisnick et al., 2019b), then the discriminative part, a softmax/cross entropy model, does not have reliable predictive uncertainty.

Results In Figure 3.8, we show a normalised histogram for the kernel distances of CIFAR-10 and SVHN. We see that most of CIFAR-10 is very close to 1, while SVHN is uniformly spread out over the range of distances. This shows that DUQ works as expected and that out of distribution data ends up away from all the centroids in feature space.

The rejection classification plot, Figure 3.9, is created similar to the previous experiment in the last section. Note that this time the *Theoretical Maximum* line is significantly lower, because the SVHN test set contains close to $26,000$ elements, while CIFAR-10's only contains $10,000$. This means that the best possible

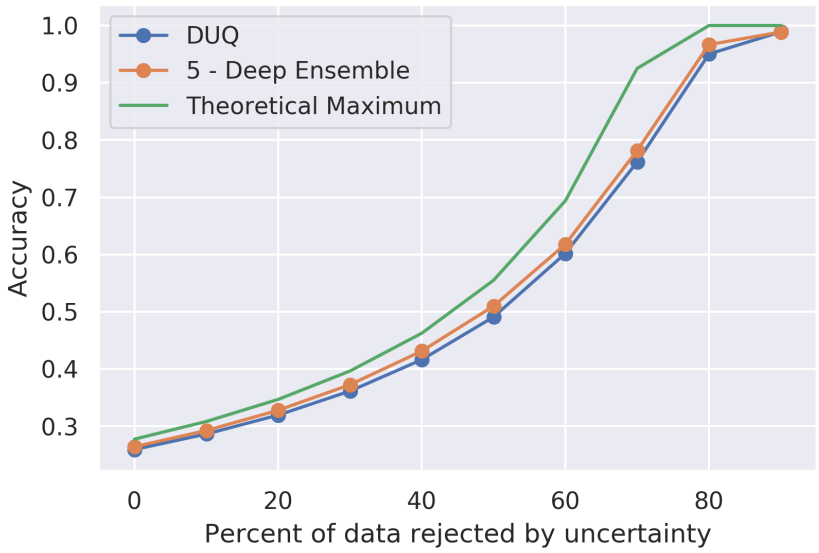


Figure 3.9: Rejection classification plot. Model performance on a mix of CIFAR-10 and SVHN, while rejecting uncertain points. The theoretical maximum is achieved when a hypothetical classifier obtains 100% accuracy on CIFAR-10 and rejects all SVHN data points first. We see that DUQ and a 5 element Deep Ensembles perform very similar.

accuracy when 100% of the data is considered is about 28%. We see that DUQ and Deep Ensembles perform similarly.

In Table 3.3, we compare DUQ with several alternative methods. We see that DUQ performs competitively with a number of recent approaches. Interestingly, on these more complicated datasets Deep Ensembles perform the best. We suspect this is because the complexity of the dataset allows the ensemble elements to be more diverse while still explaining the data well. We further see a significant gap between DUQ with and without a gradient penalty: there is a big improvement going from $\lambda = 0$ to $\lambda = 0.5$. Our hypothesis is that this is due to a lot of within class variation, which incentivises the model to collapse more diverse data points to the class centroids.

Runtime One of the main advantages of DUQ over Deep Ensembles is computational cost. For Deep Ensembles, both computation and memory cost scale linearly in the number of ensemble components, during both train and test time. DUQ has to compute the Jacobian at training time, which is expensive, but at test time there is only a marginal overhead over a softmax based model. Training

Method	AUROC
DUQ ($\lambda = 0.5$)	0.927 ± 0.013
DUQ ($\lambda = 0$)	0.861 ± 0.032
LL ratio (generative model)	0.930
Single model	0.906 ± 0.007
3 - Deep Ensembles	0.926 ± 0.010
5 - Deep Ensembles	0.933 ± 0.008
10 - Deep Ensembles	0.941
15 - Deep Ensembles	0.942

Table 3.3: We show the AUROC for separating CIFAR-10 from SVHN. Deep Ensembles is by Lakshminarayanan et al. (2017), but re-implemented and evaluated using our architecture. LL ratio is as reported in Ren et al. (2019). Single model is our architecture, but trained with softmax/cross entropy.

for one epoch on an Nvidia 1080 Ti GPU, takes 21 seconds for a softmax/cross entropy model, which implies 105 seconds for a Deep Ensemble with 5 components. DUQ with gradient penalty needs 103 seconds for one epoch at training time, which is on par with Deep Ensembles. At test time, DUQ is ~25% slower at test time than a single softmax/cross entropy model, but about 4 times faster than a Deep Ensemble with 5 components.

3.4 Deep Deterministic Uncertainty

We now take a step back and investigate whether and how the DUQ model can be simplified and improved. This section is based on work that took place after publishing the previous part of this chapter. We start by noting that in the DUQ algorithm, we set the number of centroids equal to the number of classes in practice. While it is technically possible to use more centroids, this has not shown to be of value empirically. We also note that DUQ requires numerous changes to the training procedure (including updating centroids, setting λ, a class dependent weight matrix). These changes can be cumbersome in practice, especially since the softmax output is ubiquitous and well understood. DUQ also incurs a significant training slow down compared to a standard softmax model.

To alleviate these shortcomings, we propose a simplified algorithm, *Deep Deterministic Uncertainty* (DDU), in which we train a deep model using the standard softmax output and cross entropy loss, but include a sensitivity constraint. In contrast to DUQ's gradient penalty, the sensitivity constraint is based on spectral normalisation (Miyato et al., 2018) and residual connections (Liu et al., 2020). We discuss this type of feature space restriction further in Chapter 4. After training, we compute the class-wise mean and covariance of all feature representations in the training set. We use these to define a Gaussian Mixture Model (GMM), with which we can compute the likelihood of the feature representation of a new data point. For a class mean μ_c, and covariance Σ_c, the likelihood of a data point $\boldsymbol{x}$ under the GMM is:

$$p(\boldsymbol{x}|\theta, \mathcal{D}_{\text{train}}) = \frac{1}{C}\sum_c \det(2\pi\Sigma_c)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(f_\theta(\boldsymbol{x}) - \mu_c)^\intercal \Sigma_c^{-1}(f_\theta(\boldsymbol{x}) - \mu_c)\right). \quad (3.9)$$

If we were to use the likelihood under each class for classification, then this method is the same as Gaussian Discriminant Analysis (Murphy, 2012).

We can view the likelihood under the GMM as a measure of epistemic uncertainty. It is also possible to obtain aleatoric uncertainty estimates by leveraging the softmax entropy. We discuss how to do this and how to combine the two sources of uncertainty in the next section DDU is significantly simpler to implement than DUQ and

Method	AUROC
DUQ	93.71 ± 0.61
DDU	97.86 ± 0.19
5 - Deep Ensembles	97.73 ± 0.31

Table 3.4: We show the AUROC for distinguishing CIFAR-10 from SVHN using uncertainty estimates. All models are based on the Wide-ResNet-28 (Zagoruyko et al., 2016) (in contrast to the ResNet-20 used previously this chapter).

outperforms both DUQ and Deep Ensembles, see Table 3.4. Algorithm 1 provides a description of DDU in pseudocode form and an implementation is publicly available.[4]

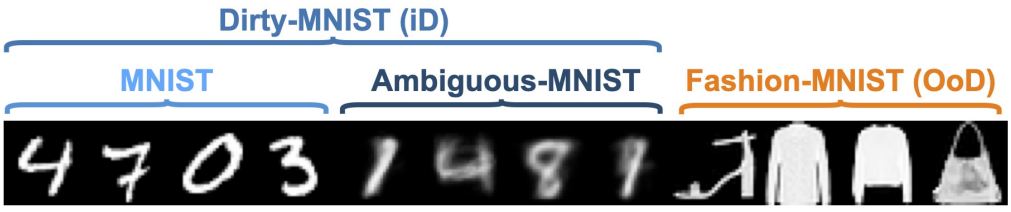
3.4.1 Disentangling the Sources of Uncertainty

In many applications of uncertainty estimation, such as active learning, and exploration in reinforcement learning (Zintgraf et al., 2021), we care specifically about finding data points with high epistemic and low aleatoric uncertainty. Meanwhile, in the uncertainty estimation literature it is common to evaluate methods on heavily curated datasets such as MNIST and CIFAR-10, which contain almost no ambiguous examples (Aitchison, 2020; Noci et al., 2021; Kapoor et al., 2022). This means it is difficult to evaluate disentangling of uncertainty using these datasets. To examine this pathology in more detail, we introduce a new dataset called Dirty-MNIST. Dirty-MNIST is a modified version of MNIST (LeCun et al., 1998a) with additional ambiguous digits (Ambiguous-MNIST) with multiple plausible labels and thus higher aleatoric uncertainty (see Figure 3.10a for several examples).

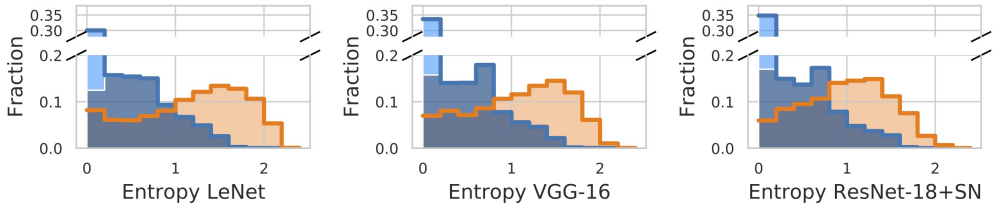
We now discuss how DDU is able to disentangle aleatoric and epistemic uncertainty. First, we note that in order to detect aleatoric uncertainty, we must have low epistemic uncertainty. Only when there is a robust understanding of each class, can an input that is ambiguous between two classes be detected (see also Section 2.3.1). In DDU, this means that the input has high likelihood under the GMM. Second, given that we are in the low epistemic uncertainty regime, we can trust the entropy of the softmax to accurately capture aleatoric uncertainty (Gal, 2016). In Figure 3.11, we show the effect of disentangling epistemic and

[4] `https://github.com/omegafragger/DDU`

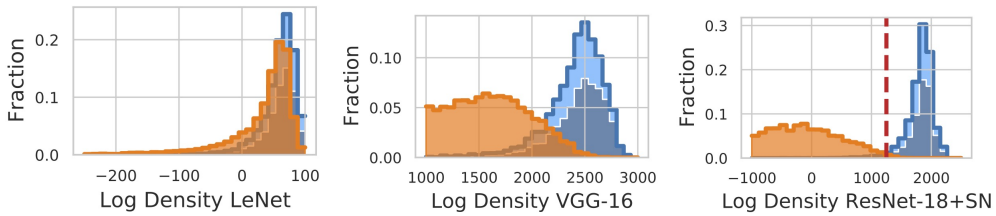
aleatoric uncertainty in practice in an active learning setup. Using DDU or a Deep Ensemble (Lakshminarayanan et al., 2017) with BALD (Houlsby et al., 2011) performs significantly better than methods that measure total uncertainty.



(a) Dirty-MNIST (iD) and Fashion-MNIST (OOD)

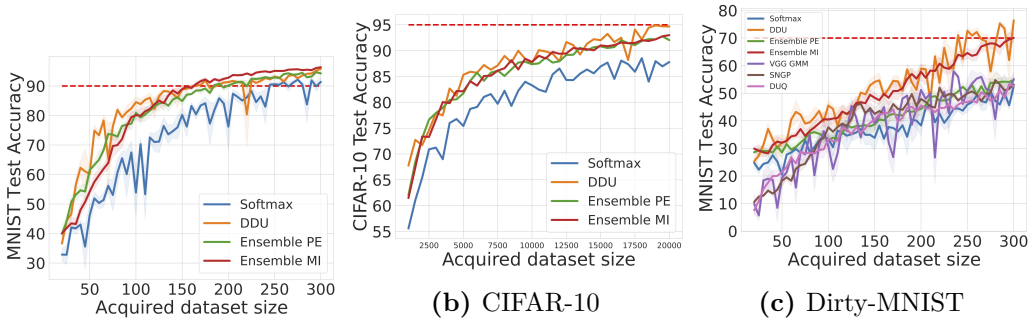


(b) Softmax entropy



(c) Feature-space density

Figure 3.10: In the bottom row (Figure c), we see that with a well-regularised feature space (DDU with ResNet-18+SN), iD (Dirty-MNIST) and OOD (FashionMNIST) 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. In all cases, unambiguous and ambiguous iD samples are confounded as their densities overlap. In the middle row (Figure b), we see that Softmax entropy captures aleatoric uncertainty for iD data (Dirty-MNIST), thereby separating standard MNIST samples and Ambiguous-MNIST samples. Note the break in the y-axis, with a significantly higher MNIST count at entropy close to 0. However, iD and OOD are confounded: softmax entropy can have low values for OOD, making it indistinguishable from iD.



(a) MNIST **(b)** CIFAR-10 **(c)** Dirty-MNIST

Figure 3.11: Active Learning experiments with DDU. We show acquired training set size vs test accuracy. DDU performs on par with Deep Ensembles.

Algorithm 1 Deep Deterministic Uncertainty

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

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

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

  if low density $q(z)$ **then**
    return OOD
  else if high density $q(z)$ **then**
    if high entropy $H_p[Y|x]$ **then**
      return ambiguous iD
    else if low entropy $H_p[Y|x]$ **then**
      return iD
    end if
  end if
end function

3.5 Conclusion

We introduced two uncertainty estimation methods: DUQ, an RBF network that is trained end-to-end, and DDU, which fits a GMM in feature space after training. By carefully considering the necessary restrictions that need to be placed on the model for uncertainty estimation, we showed that these methods outperform alternatives while being computationally efficient to evaluate.

In practice, one should consider using DDU before DUQ. DDU is simpler to implement, outperforms DUQ on large datasets, and has the ability to disentangle aleatoric uncertainty when epistemic uncertainty is low. Reasons to consider DUQ are that it can train with arbitrary architectures and can use multiple centroids per class, to account for multi-modal class distributions.

DUQ and DDU have several limitations and the most notable one is that they cannot be used for regression. In regression, there is no concept of a class, so the centroids cannot be computed. It is also not possible to compute joint uncertainty over a batch of inputs with either method. Several promising methods have already built on DUQ and DDU with the aim of overcoming some of these limitations, such as SNGP (Liu et al., 2020) and NUQ (Kotelevskii et al., 2022). We introduce our own follow-up method and compare it to SNGP in the next chapter.

4

Feature Collapse in Deep Kernel Learning

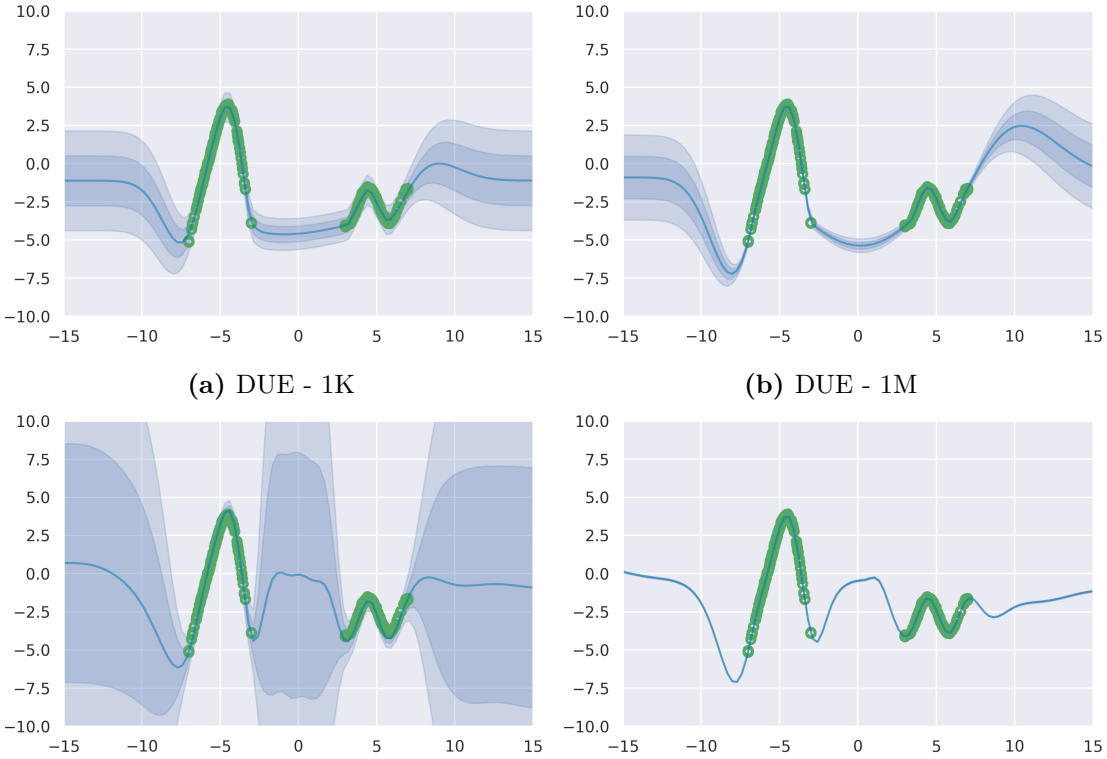
Contents

Gaussian processes (GPs) are often considered a gold standard in uncertainty estimation on low dimensional and small datasets (Rasmussen et al., 2006). Using approximate inference, it is possible to scale GPs to larger datasets while retaining many of the uncertainty estimation properties. However, the available stationary kernels do not compute a meaningful distance on high dimensional structured data, such as images. Deep Kernel Learning (DKL) was introduced as a solution to this problem: the input data is transformed using a deep feature extractor and

an approximate GP is placed on top (Hinton et al., 2008; Calandra et al., 2016; Wilson et al., 2016a). This approach has some similarities with DUQ of the previous chapter: a GP with a stationary kernel is also a distance-aware output and the data is transformed by a feature extractor in both cases. In this chapter, we focus on evaluating and improving the uncertainty estimation ability of DKL.

There are two key approaches to approximate inference in a GP on a large dataset: Random Fourier Features (RFF) (Rahimi et al., 2008; Liu et al., 2020) and variational inducing points (Titsias, 2009; Hensman et al., 2015). The RFF approximation is computationally fast, but requires infinitely many features to exactly approximate the GP posterior. The more recent variational inducing point approximation (Titsias, 2009; Hensman et al., 2015) is guaranteed to monotonically improve the approximation as the number of inducing points grow. A discussion and potential merging of these two methods is done in Hensman et al. (2017). In combination with DKL, it can obtain accuracies matching standard softmax neural networks (Wilson et al., 2016b; Bradshaw et al., 2017). We discuss the differences between the RFF and inducing point approximation in detail in Section 4.3.1. Previous works on DKL have mostly been evaluated in terms of accuracy and robustness to adversarial examples (Wilson et al., 2016b; Bradshaw et al., 2017). In this chapter, we focus on evaluating and improving the uncertainty estimation quality of DKL.

We study why previous DKL works can have poor uncertainty estimates, and propose Deterministic Uncertainty Estimation (DUE - pronounced “Dewey”), which builds on DKL and addresses this limitation. Examining why DKL uncertainty underperforms, we find that for certain feature extractors, data points dissimilar to the training data (also called “out of distribution” or OOD data) might be mapped closed to feature representations of in distribution points. This is called “feature collapse” (Figure 3.2), and suggests that a constraint must be placed on the deep feature extractor. To understand what constraints must be placed on the DKL feature extractor, we take inspiration from DUQ and SNGP (Liu et al., 2020) which propose to use a bi-Lipschitz constraint on a feature extractor in the



(a) DUE - 1K **(b)** DUE - 1M

(c) SNGP - 1K **(d)** SNGP - 1M

Figure 4.1: In green 300 example training data points and in blue the prediction including uncertainty (one and two std). We see that DUE performs well when trained with 1 thousand (1K) data points and 1 million (1M) data points. Meanwhile, the RFF approximation in SNGP concentrates its uncertainty at 1M, and is very uncertain at 1K. This highlights a drawback of the parametric RFF approximation.

context of radial basis function (RBF) networks and RFF GPs. This constraint enforces the feature representation to be sensitive to changes in the input (lower Lipschitz, avoids feature collapse) but also generalise due to smoothness (upper Lipschitz). We show the effect of these constraints in Figure 4.2.

Compared to alternative single forward pass uncertainty methods, DUE has a number of advantages. DUQ's one-vs-all loss cannot be extended to regression, while DUE is demonstrated to work well on both classification and regression. SNGP uses the RFF approximation, which means the uncertainty concentrates in the limit of data, even "far-away" from the training data. For example, in the region $-3 < x < 3$ of Figure 4.1 we are outside the support of the training data. We can see for a training set size of 1K the uncertainty interval is wide for SNGP (Figure 4.1c). However, as we increase the training set size to 1M, the uncertainty interval

outside the support of the training data becomes indistinguishable from inside the support (Figure 4.1b). In contrast to SNGP, DUE's inducing point GP preserves the exact GP's properties and has similar uncertainty outside the support of the training data when trained on small and very large datasets. This means DUE fulfils optional desideratum 4, uncertainty should remain high away from the training data. In contrast to existing DKL methods, DUE avoids feature collapse (Table 4.1), and training DUE is substantially simplified: there is no need to pre-train with a softmax output or adapt the architecture of standard models. There is also no computational overhead over a standard softmax model.

DUE outperforms all alternative single forward pass uncertainty methods on the CIFAR-10 vs SVHN detection task. To evaluate DUE's uncertainty quality for regression on real world data, we use a recently introduced benchmark for predicting treatment effect and uncertainty based treatment deferral in causal models for personalised medicine (Jesson et al., 2020). This benchmark combines the need for accurate predictions and uncertainty in a regression task, and we show that DUE outperforms all alternative methods. The code for this chapter is publicly available.[1]

In summary, our contributions are as follows:

1. A single forward pass uncertainty model that works well for classification and regression.

2. Insight into DKL's failures and a principled solution that gives accurate uncertainty estimates.

3. A practical DKL training setup that can learn the deep model and GP from scratch on a variety of datasets.

4.1 Background

In DKL, a deep feature extractor is used to transform the inputs over which an inducing point GP is defined. DKL was originally introduced as a way to combine

[1] `https://github.com/y0ast/DUE`

the expressiveness of deep neural networks with the probabilistic prediction ability of GPs. The kernel, which contains a deep feature extractor, is defined as:

$$k_{l,\theta}(\boldsymbol{x}_i, \boldsymbol{x}_j) \to \bar{k}_l(f_\theta(\boldsymbol{x}_i), f_\theta(\boldsymbol{x}_j)), \tag{4.1}$$

where $f_\theta(\cdot)$ is a deep neural network, such as a Wide ResNet (WRN) (Zagoruyko et al., 2016) up to the last linear layer, parametrised by θ. The base kernel $\bar{k}_l(\cdot, \cdot)$ can be any of the standard kernels, such as the RBF or Matérn kernel. l represents the hyperparameters of the base kernel, such as the length scale and output scale. $\boldsymbol{x}_i$ is a data point in the dataset $\mathcal{D}_{\text{train}} = \{\boldsymbol{X}, \boldsymbol{Y}\}$. DKL can be applied to both classification and regression problems.

Inference in an exact GP is bottlenecked by the inversion of an $n \times n$ kernel Gram matrix with n the number of data points, which has a time complexity that scales cubically with n. In contrast, an inducing point GP is based on m inducing points (with $m << n$), which act as pseudo-input-points used to approximate the full dataset. The locations and values of the inducing points are parameters, and learned by maximising a lower bound on the marginal likelihood known as the ELBO (Titsias, 2009; Hensman et al., 2015). This reduces the complexity of posterior inference from $\mathcal{O}(n^3)$ to $\mathcal{O}(m^2 n)$, thus models with fewer inducing points are faster to train. In practice, these inducing points are placed in feature space to exploit the clustering behaviour of the deep feature extractor, we visualise this behaviour in the Figure 4.4.

The two most prevalent instances of DKL are SV-DKL (Wilson et al., 2016b) and GPDNN (Bradshaw et al., 2017). In SV-DKL, an additional restriction is placed on the inducing points to lie on a grid. This enables faster matrix inversion algorithms, at the expense of a less flexible inducing point structure. In GPDNN, the inducing points are not constrained, but the feature dimensionality is reduced to just 25, leading to increased risk of feature collapse and poor uncertainty estimation (see Section 4.2.1). Both methods use a pre-training phase with a standard, non GP, output. GPDNN trains with different optimisers for pre-training and training with GP output, and uses different learning rates for variational versus model parameters.

In summary, these models are more cumbersome to use than alternative methods for uncertainty estimation, such as Deep Ensembles (Lakshminarayanan et al., 2017).

4.2 Method

Here we explain how feature collapse affects DKL's uncertainty, give insight into why DKL's objective leads to feature collapse, and use that insight to mitigate feature collapse by introducing the DUE model.

4.2.1 Feature Collapse

When the deep feature extractor inside the kernel is unconstrained, it can map in and out of distribution data to the same location in feature space. The GP assigns high confidence to inputs similar to the training data, so when in and out of distribution data have the same feature representation, the GP will assign high confidence to out of distribution inputs. This is called *feature collapse*, which we previously discussed in Chapter 3 and visualised in Figure 3.2.

A compelling explanation for why feature collapse is likely to occur for DKL in particular is given in Lotfi et al. (2022), who argue that the issue lies with the log marginal likelihood objective. The authors show that an unconstrained feature extractor in DKL can lead to severe overfitting and incorrect uncertainty and the problem is exacerbated where there are a large number of hyperparameters to learn as in DKL. In the low-data regime, the authors show it is possible to significantly outperform the marginal likelihood objective using a conditional log marginal likelihood (CLML) objective. We show how this approach is complimentary with DUE in Appendix B.1.

Following Chapter 3, we know that feature collapse can be reduced by enforcing two constraints on the model: sensitivity and smoothness. Sensitivity implies that when the input changes the feature representation also changes. Thus, the model cannot simply collapse feature representations arbitrarily. Smoothness implies small changes in the input cannot cause massive shifts in the output. These constraints can be achieved by enforcing the feature extractor to be bi-Lipschitz, which has also been

Algorithm 2 Algorithm for training DUE.

1: **Definitions:**

1.1: Residual NN $f_\theta : x \to \mathbb{R}^J$ with feature space dimensionality J and parameters θ.

1.2: Approximate GP with parameters $\phi = \{l, s, \omega\}$, where l length scale and s output scale of $\bar{k}$, ω GP variational parameters (including m inducing point locations Z).

1.3: Learning rate η, loss function $\mathcal{L}$.

2: Using a random subset of p points of our training data, $X^{\text{init}} \subset X$, compute:

2.1: **Initial inducing points:** K-means on $f_\theta(X^{\text{init}})$ with $K = m$. Use found centroids as initial inducing point locations Z in GP.

2.2: **Initial length scale:** $l = \frac{1}{\binom{p}{2}} \sum_{i=0}^{p} \sum_{j=i+1}^{p} |f(X_i^{\text{init}}) - f(X_j^{\text{init}})|_2^2$.

3: **for** Minibatch $\boldsymbol{x}_b, \boldsymbol{y}_b \subset \mathcal{D}_{\text{train}}$ **do**

4: $\theta' \leftarrow$ spectral_normalisation(θ) – explained in Section B.1.5.

5: $p(\boldsymbol{y}'_b|\boldsymbol{x}_b) \leftarrow$ evaluate_GP$_\phi(f_{\theta'}(\boldsymbol{x}_b))$

6: $\mathcal{L} \leftarrow \text{ELBO}_\phi(p(\boldsymbol{y}'_b|\boldsymbol{x}_b), \boldsymbol{y}_b)$

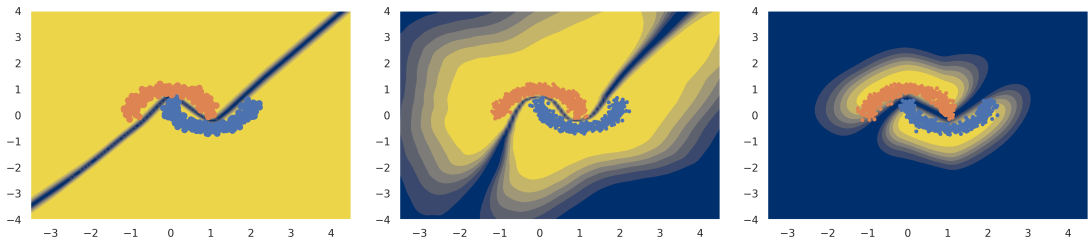
7: $(\phi, \theta) \leftarrow (\phi, \theta) + \eta * \nabla_{\phi,\theta}\mathcal{L}$

8: **end for**

shown to help in many other contexts (Rosca et al., 2020). A bi-Lipschitz feature extractor can be obtained in different ways and each comes with different trade-offs:

- **Two-sided gradient penalty:** this penalty was introduced in the context of GANs (Gulrajani et al., 2017). It regularises using a two-sided gradient penalty, penalising the squared distance of the gradient from a fixed value at every input point. It is easy to implement, but only a soft constraint. In practice the training stability and regularisation effectiveness is sensitive to the relative weight of the penalty in the loss. This method is used in DUQ.

- **Direct spectral normalisation and residual connections:** spectral normalisation (Gouk et al., 2018; Miyato et al., 2018) on the weights leads to smoothness, and it is possible to combine this with an architecture that contains residual connections for the sensitivity constraint. This method is faster than the gradient penalty, and in practice offers a more effective way of mitigating feature collapse. This method is used in SNGP (Liu et al., 2020).



(a) ResNet + Softmax **(b)** GPDNN **(c)** DUE

Figure 4.2: We show uncertainty results on the two moons dataset. Yellow indicates high confidence, while blue indicates uncertainty. In Figure 4.2a, a simple feed-forward ResNet with a softmax output is certain everywhere except on the decision boundary. In Figure 4.2b, we see that GPDNN (Bradshaw et al., 2017), which uses a simple Feed-Forward Network as feature extractor, is certain even far away from the training data. In Figure 4.2c, we show DUE, which has the appropriate restrictions on the feature extractor (residual connections and spectral normalisation) and obtains close to ideal uncertainty on this dataset.

- **Reversible model:** A reversible model is constructed by using reversible layers and avoiding any down scaling operations (Jacobsen et al., 2018; Behrmann et al., 2019). This approach can guarantee that the overall function is bi-Lipschitz, but it consumes considerably more memory and can be difficult to train.

In this work, we use direct spectral normalisation and residual connections, as we find it to be more stable than a direct gradient penalty and significantly more computationally efficient than reversible models. In Figure 3.2, we show that a constrained model does not collapse points on top of each other in feature space, enabling the GP to correctly quantify uncertainty. Using a stationary kernel, the model reverts to the prior away from the training data just like a GP defined over the input space.

4.2.2 Model

DUE builds upon the foundation of GPDNN (Bradshaw et al., 2017). To enforce the bi-Lipschitz constraint and enable high quality uncertainty, we restrict the deep feature extractor to have residual connections in combination with spectral normalisation as described in the previous section. We further make several

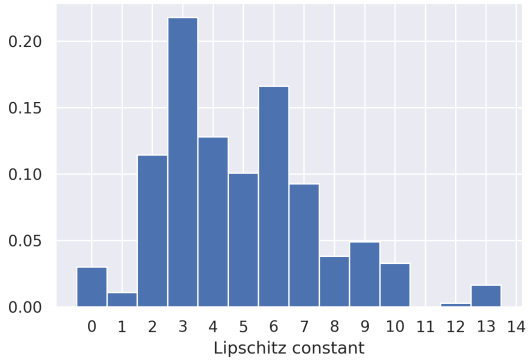


Figure 4.3: A density of the Lipschitz values in batch normalisation layers, averaged across 15 WRN models that were trained with Softmax output and without spectral normalisation (exactly following Zagoruyko et al. (2016)). We see that many of the constants are significantly above 1, highlighting that batch normalisation has significant impact on the Lipschitz constant of the network.

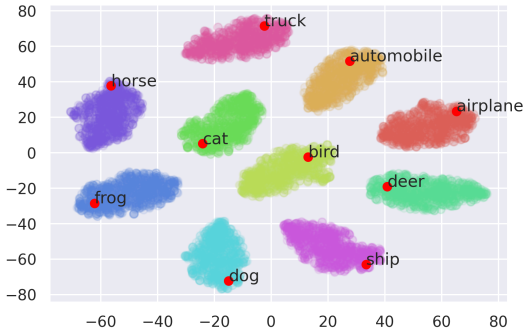


Figure 4.4: Visualisation of the learned feature representations of the training set of CIFAR-10. We use T-SNE (van der Maaten et al., 2008) to reduce the dimensionality down to 2D, and we colour the points by their class label. We overlay (in red) the inducing points location, which we label by computing the closest class average T-SNE representation. The features and inducing point locations are obtained from a DUE model trained with 10 inducing points and spectral normalisation.

simplifications to the training process to make GPDNN more practical to use. Instead of an additional downsampling layer to 25 dimensions which increases the risk of feature collapse, the GP output is placed directly on top of the last convolutional layer of a large model, which is 640 dimensional in the case of the WRN. In practice, DUE trains well with just 10 inducing points on CIFAR-10 which leads to a runtime that is only 3% slower than a softmax model. No pre-training is necessary and training is stable with a single set of hyperparameters for both the variational and model parameters. The training procedure is described in Algorithm 2.

Online power iteration for spectral normalisation can be implemented exactly for fully connected layers and 1x1 convolutions. For convolutions larger than 1x1, we use an approximate method that lower bounds the exact method, as proposed in Gouk et al. (2018) and implemented by Behrmann et al. (2019).

Batch Normalisation In SNGP, spectral normalisation is applied only on the convolution operations. However, batch normalisation, a crucial component of

Table 4.1: AUROC on CIFAR10 vs SVHN for two DKL methods trained with feature extractors with and without a bi-Lipschitz constraint. Non bi-Lipschitz uses VGG-19 (Simonyan et al., 2015); bi-Lipschitz uses a WRN with spectral normalisation. Both models obtain high accuracy, matching standard softmax models. SV-DKL (Wilson et al., 2016b) without bi-Lipschitz obtains poor uncertainty: distinguishing in and out of distribution data is no better than chance. GPDNN (Bradshaw et al., 2017) without bi-Lipschitz obtains better but still poor uncertainty. The cell highlighted in grey is DUE.

	Bi-Lipschitz		non bi-Lipschitz	
	AUROC	Accuracy	AUROC	Accuracy
SV-DKL	**0.959±.001**	95.7±0.06	0.498±.001	93.6±0.05
GPDNN	**0.958±.005**	95.6±0.04	0.876±.004	93.7±0.10

training deep models, has a non-trivial Lipschitz constant. Batch normalisation transforms the input following:

$$x_{out} = \text{diag}\left(\frac{\gamma}{\sqrt{\text{Var}(x)}}\right)(x - \mathbb{E}[x]) + \beta, \tag{4.2}$$

with γ and β the learnable scale and shift parameters. It has a Lipschitz constant of $\max_i |\frac{\gamma_i}{\sqrt{\text{Var}(x)}_i}|$ (Gouk et al., 2018). Using the above equation, we can extend spectral normalisation to batch normalisation by dividing the weight γ of the batch normalisation by the (scaled) Lipschitz constant. In practice, we find that batch normalisation layers in trained ResNets have a relatively high Lipschitz constant, up to around 12, and 95% of the channel-wise Lipschitz constants are greater than one (see also Figure 4.3). Adding spectral normalisation to batch normalisation layers ensures the entire network's upper Lipschitz constant is bounded.

4.3 Related Work

The single forward pass uncertainty methods most similar to DUE are DUQ and SNGP (Liu et al., 2020). Compared to DUQ, DUE can readily be applied to regression problems, and the gradient penalty in DUQ is difficult to optimise and slow to compute. SNGP (Liu et al., 2020) consists of a deep feature extractor and an output GP, which is approximated using the parametric Random Fourier Features (RFF) approximation (Rahimi et al., 2008) in combination with a Laplace approximation for the non-conjugate Softmax likelihood (Rasmussen

et al., 2006). We discuss the trade-offs between the RFF and variational inducing point approximations in the next subsection.

4.3.1 Inducing Points versus RFF Approximation

With the RFF approximation, for any *finite* number of features, the kernel is approximated as a linear model on a finite number of features. Inducing point GP approximations, on the other hand, use as the approximation a standard GP which is defined over a set of *inducing inputs* instead of over the entire training set (which is computationally prohibitive). The inducing point approximation then tries to move the inducing input locations to minimise the KL between this inducing point GP and the full GP (Titsias, 2009), giving a tractable objective. The approximation will become exact when the number of inducing points matches the number of training points (at which point the ELBO can match the GP marginal likelihood by placing the inducing inputs on the training input locations). This is in contrast to RFF which will only become exact at the infinite limit of the number of random features. Inducing point GPs also have a tight bound on the marginal log-likelihood, which means that we can do model selection by optimising the ELBO with respect to various hyperparameters (Burt et al., 2019).

In Figure 4.1, we show the effects of the approximations in DUE and SNGP in practice by training on a small and large dataset sampled from the same distribution. SNGP's uncertainty interval is dependent on the number of training points: it is wide in areas where there is no training data at 1K data points, but very narrow in the same regions at 1M data points. While DUE's inducing point GP has similar uncertainty outside the support of the training data when trained on either dataset size. SNGP uncertainty was estimated using the exact method with a ridge penalty of 1. All other hyperparameters were shared between the two methods, and are listed in Appendix B.1.

Table 4.2: Results on the CIFAR-10 dataset, and distinguishing between CIFAR-10 and SVHN by uncertainty. All results use a WRN (Zagoruyko et al., 2016) as feature extractor and are the average and standard error of 5 runs. Ensemble of 5 by Lakshminarayanan et al. (2017), DUQ, SNGP by Liu et al. (2020), SV-DKL (with the same constraints as DUE - spectral normalisation and residual connections) by Wilson et al. (2016b). The runtime is relative to the "Standard WRN" row. In bold are top results (within standard error), and the horizontal line separates ensembles from single forward pass methods.

Method	Runtime ↓	Accuracy (%) ↑	AUROC ↑
Ensemble of 5	5x	**96.6±0.03**	**0.967±0.005**
Standard WRN	**1x**	**96.2±0.01**	0.932±0.008
DUQ	3.5x	94.9±0.04	0.940±0.003
SNGP	**1x**	96.0±0.04	0.940±0.006
SV-DKL (with constraints)	2x	95.7±0.06	**0.959±0.001**
DUE	**1x**	95.6±0.04	**0.958±0.005**

4.4 Experiments

We now highlight in several experiments how the suggested constraints affect uncertainty estimation in DKL. The first experiment looks at classification in 2D, and subsequently we scale the model to CIFAR-10. Lastly, we look at an application of DUE in a real-world setting.

4.4.1 Feature Collapse in DKL

We analyse the problem of feature collapse in DKL in Figure 4.2 on the Two Moons dataset (Pedregosa et al., 2011). We show results for three different models: a standard softmax model, DUE, and a variation where the spectral normalised ResNet is replaced by a fully connected model (similar to Bradshaw et al. (2017)), experimental details are provided in Appendix B.1. The uncertainty is computed using the predictive entropy of the class prediction; we model the problem as a two class classification problem.

The standard softmax output model is certain everywhere except on the decision boundary. DUE (Figure 4.2c) quantifies uncertainty as expected for the two moons dataset: certain on the training data, uncertain away from it and in-between the two moons. Figure 4.2b highlights the importance of our contribution, because the

Table 4.3: Test accuracy and Negative Log-Likelihood (NLL) on CIFAR-10 of DUE with WRN feature extractor for increasing number of inducing points (m). As the number of inducing points increases, both the NLL and accuracy remain constant with no statistically significant difference. This shows that is not necessary to have many inducing points to obtain strong performance.

m	Accuracy (%) ↑	NLL ↓
10	95.56±0.04	0.187±0.004
50	95.54±0.06	0.182±0.002
100	95.35±0.06	0.183±0.002
1000	95.49±0.05	0.180±0.001

Table 4.4: ECE calibration results on CIFAR-10 with 15 bins and no post-hoc scaling. DUQ and SNGP results obtained from Liu et al. (2020).

Method	ECE
DUE	0.018±0.002
DUQ	0.034±0.002
SNGP	0.018±0.001

uncertainty estimation of a standard Feed-Forward Network in combination with DKL suffers from feature collapse and is certain away from the training data.

4.4.2 Feature Collapse in CIFAR-10 versus SVHN

We now consider training end-to end with a large feature extractor, the Wide Residual Network (WRN), on CIFAR-10 (Krizhevsky et al., 2009). We follow Zagoruyko et al. (2016) and use a 28 layer model with BasicBlocks and dropout. Importantly, we can use their hyperparameters (such as dropout rate, learning rate, and optimiser) when training DUE, and no further tuning is necessary. We remove the final linear layer of the original WRN and the 640-dim feature vector is directly used in the GP. We use 10 inducing points for DUE on CIFAR-10, which leads to a runtime of 1m47s for one epoch on an Nvidia GTX1080 Ti, while a standard WRN with a linear+softmax output takes 1m43s. An ablation were we look at the learned location of the 10 inducing points is available in the Figure 4.4. Figure 4.4 is created by transforming all points in the CIFAR-10 training dataset into their feature representation using the trained feature extractor. We then use T-SNE (van der Maaten et al., 2008) on all the feature representations combined and reduce them to two dimensions. We plot all points coloured by their class and overlay the

Table 4.5: Comparing the performance in terms of RMSE of several treatment effect and uncertainty estimation models under "random" and "uncertainty" based deferral to an expert (50% and 10% deferred for IHDP Cov. and IHDP respectively). The first three rows were obtained from Jesson et al. (2020), DKLITE (Zhang et al., 2020) was evaluated using the author's open source implementation and the ensemble of TARNet was reimplemented. BART by Chipman et al. (2010), BTARNet by Shalit et al. (2017), BCEVAE by Louizos et al. (2017). DUE outperforms all alternative methods, while being 5x faster to train and evaluate than the ensemble.

	IHDP Cov. (50% def.)		IHDP (10% def.)	
Method	*random*	*uncertainty*	*random*	*uncertainty*
BART	2.6±.2	1.8±.2	1.90±.20	1.60±.10
BTARNet	2.2±.3	1.2±.1	1.10±.03	0.76±.03
BCEVAE	2.5±.2	1.7±.1	1.80±.06	1.47±.08
DKLITE	2.6±.7	1.8±.5	1.74±.53	1.34±.41
Ensemble of 5 TARNet	1.74±.1	1.19±.03	1.14±.04	0.76±.01
DUE	**1.63±.06**	**1.05±.05**	**0.91±.04**	**0.48±.02**

inducing point locations in red. For each inducing point, we add a label based on the class of the closest feature representation in the training set.

To further investigate the surprisingly strong performance with just 10 inducing points, we train DUE using more inducing points and present the results in Table 4.3. As expected, the NLL improves as the number of points is increased, but the difference is small. We attribute this surprising result to the flexibility of deep learning and the quality of the WRN.

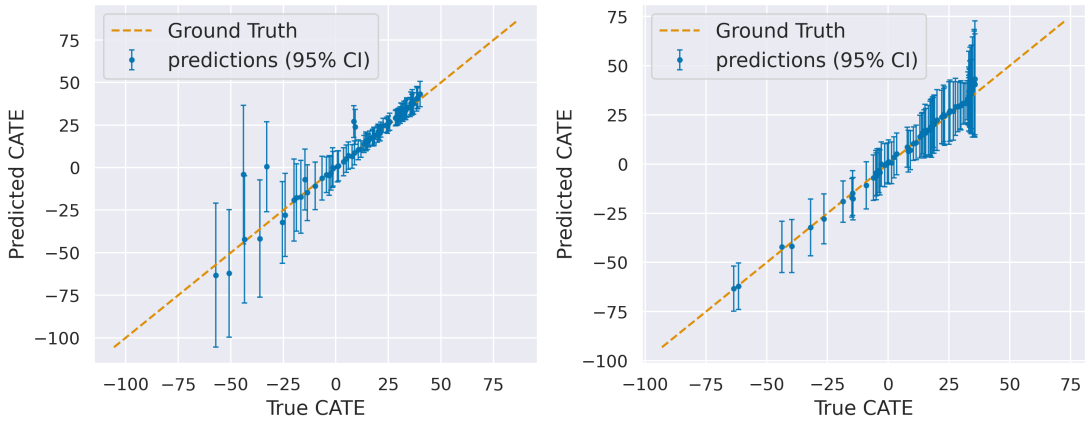
Uncertainty To compare the uncertainty quality between different models, we use the experiment of distinguishing between the test sets of CIFAR-10 and SVHN (Netzer et al., 2011) – a notably difficult dataset pair (Nalisnick et al., 2019a) – using each model's uncertainty metric. In Table 4.1, we compare SV-DKL and GPDNN with two different feature extractors: with and without the DUE constraints. In contrast to the original SV-DKL and GPDNN, these models were trained from scratch using the same hyperparameters (including mini-batch-size) as DUE. Note that GPDNN with constraints and the improved training setup is conceptually the same as DUE. We see that in both cases the accuracy is high, but the uncertainty is very poor when no constraints are placed. This highlights the importance of

our contribution: additional constraints on the feature extractor are crucial for good uncertainty performance in DKL.

Table 4.2 compares DUE to several baselines. We train DUE using 10 inducing points and spectral normalisation constant 3, set in similar fashion to Liu et al. (2020) by choosing the lowest value that does not significantly affect accuracy. All methods compute uncertainty using the predictive entropy, except DUQ which uses the closest kernel distance. We see that DUE and SV-DKL, with DUE feature extractor, outperform all competing single forward pass uncertainty methods. Recall, the difference between DUE and SV-DKL is that DUE learns the inducing points jointly with the model, but SV-DKL constrains them to lie on a grid to enalbe faster matrix inversion algorithms at the expensive of a less flexible inducing point structure. DUE is competitive with Deep Ensembles uncertainty, but only requires a single forward pass and is therefore five times faster to train. SV-DKL with the DUE feature extractor performs similar to DUE, but takes 2x longer to train, since it needs to invert a larger covariance matrix. The DUE and SV-DKL result shows that the theoretically preferable inducing point variational approximation also leads to practical improvements, as both outperform SNGP in terms of uncertainty estimation.

4.4.3 Uncertainty in Regression for Personalised Healthcare

To demonstrate the performance of DUE on a regression task, we focus on a new benchmark for personalised healthcare (Jesson et al., 2020). The task is to predict responses of individuals to a particular treatment, which is only possible if there is sufficient data available to assess how they might respond. This is also called the *overlap assumption*, which states that to predict the effect of treatment for a particular input x (an individual described by features), we need to have seen similar data points that have received treatment as well as points that have not received treatment. We can use uncertainty to assess when this assumption is violated and the patient should be referred to an expert instead (Jesson et al., 2020). It is important that the uncertainty estimates are accurate, otherwise an



(a) BTARNet **(b)** DUE

Figure 4.5: Predicted CATE versus true CATE with 95% confidence intervals for a randomly chosen cross-validation run. DUE is confident (without converging to no uncertainty) and correct, while BTARNet (Shalit et al., 2017) is wrong in 2 instances and the true CATE is not within the confidence interval.

individual could receive treatment even if their response to treatment is not known which can result in undue stress, financial burdens, or worse.

This benchmark assesses both predictive performance and uncertainty estimation. Other benchmarks which only evaluate test log-likelihood, such as UCI (Dua et al., 2017), only capture uncertainty on the test set (in distribution), which is insufficient to evaluate uncertainty on out of distribution data. In the case of feature collapse, the test log-likelihood is therefore not a useful comparison tool.

We train DUE on the input features of the training set, appending a 0 or 1 to represent no treatment and treatment. The model is then used to predict the Conditional Average Treatment Effect (CATE) (Abrevaya et al., 2015), computed as the difference between the expected effect of treatment and no treatment. The CATE estimate and its uncertainty are the expectation and variance over the difference in joint predictions with the two inputs:

$$[y_0, y_1] \sim \text{DUE}([\boldsymbol{x}, t = 0], [\boldsymbol{x}, t = 1])$$

$$\text{CATE}(x) = \mathbb{E}[y_1 - y_0] \qquad \text{CATE}_\text{u}(x) = \text{Var}[y_1 - y_0].$$

The CATE can be computed exactly, using the mean of the GP posterior, while we use Monte Carlo sampling of the joint posterior for the uncertainty (note that

this requires only a single forward pass through the model, and sampling from the GP is fast). We use the uncertainty to decide which predictions will be deferred to an expert. This process allows us to make a *causal* statement on the effect of treatment, under the assumptions in Jesson et al. (2020).

We use the IHDP (Hill, 2011) and IHDP Covariate shift (referred to as IHDP Cov.) datasets. IHDP is a regression dataset with ∼750 data points, and IHDP Cov. is a variant with additional covariate shift to increase the difficulty of the task. These are real world datasets derived from the Infant Health and Development Program. The details of the covariate shift are discussed in Jesson et al. (2020). In the experiments, treatment-effect recommendations are deferred to an expert if the CATE estimate has high uncertainty. We include a baseline that defers at random. We run IHDP and its covariate shift variant for 1,000 cross-validation trials.

In Figure 4.5, we compare DUE with Bayesian TARNet, which is the standard TARNet (Shalit et al., 2017) extended with MC dropout (Gal et al., 2016) for uncertainty quantification. The TARNet is a commonly used deep learning baseline in the causality field. The results show that DUE is more accurate than BTARNet for most CATE values, and that the BTARNet makes predictions for which the ground truth is not within the confidence interval. Table 4.5 summarises our results compared to several baselines (detailed in Appendix B.1.3) and shows that DUE has improved performance and uncertainty estimates better suited to rejection policies than other uncertainty-aware methods.

4.5 Active Learning for Personalised Healthcare

We now extend the previous section for active learning (AL). This section is based on work that took place after publishing the previous part of this chapter. In standard AL, we care about labelling only the most informative data points, see also Section 2.4, but in the context of personalised healthcare we additionally care about satisfying the overlap assumption.

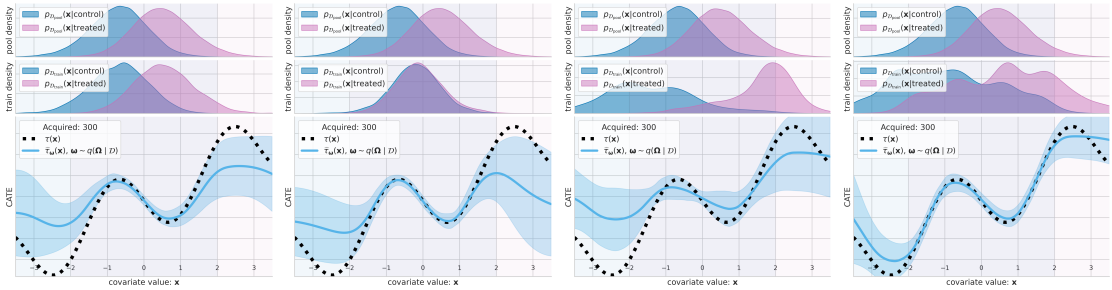
To motivate Causal–BALD, we first look at a set of simple acquisition functions. A random acquisition function selects data points uniformly at random from $\mathcal{D}_{\text{pool}}$ and adds them to $\mathcal{D}_{\text{train}}$. In Figure 4.6a, we have acquired 300 such examples from a synthetic dataset and trained DUE on those labelled examples. Comparing the top two panels, 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 panel, we see that while the CATE estimator is accurate and confident near the modes of $\mathcal{D}_{\text{pool}}$, it becomes less accurate as we move to lower-density regions. In this way, the random acquisition of data 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 the propensity score, the probability of observing a treated and untreated individual, to bias data acquisition toward regions where the overlap assumption is satisfied.

Definition 4.5.1. *Counterfactual Propensity Acquisition*

$$\mathrm{I}(\widehat{\pi}_{\mathrm{t}} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}_{\text{train}}) \equiv 1 - \widehat{\pi}_{\mathrm{t}}(\boldsymbol{x}) \tag{4.3}$$

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 $\mathbf{X}$, individuals described by features, and $\mathbf{T}$, if they underwent treatment, so it is straightforward to train an estimator for the propensity, $\widehat{\pi}_{\mathrm{t}}(\boldsymbol{x})$. Figure 4.6b shows that while propensity score acquisition matches the treated



(a) Random **(b)** Propensity **(c)** τ–BALD **(d)** μ–BALD

Figure 4.6: Simple acquisition functions. In all cases, we run active learning for a fixed budget of 300 acquisitions. In the case of Random, we see that as expected the first two rows are very similar and there is no noticeable bias. For Propensity, there is virtually full overlap at the expensive of diversity. For τ–BALD, the dataset spans the full domain at the expense of very limited overlap. μ–BALD performs relatively better: there is overlap and diversity. This leads to a confident estimate of the CATE for the full domain of x.

and control densities in the train set, it still biases data selection towards the modes of $\mathcal{D}_{\text{pool}}$.

We now look at adapting the BALD acquisition function to this new setting. The goal of BALD is to acquire data $(\boldsymbol{x}, \mathrm{t})$ that maximally reduces uncertainty in the model parameters θ 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 4.5.2. *τ–BALD*

$$\mathrm{I}(\mathrm{Y}^1 - \mathrm{Y}^0; \theta \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}_{\text{train}}) \approx \underset{\boldsymbol{w} \sim p(\theta \mid \mathcal{D}_{\text{train}})}{\mathrm{Var}} \left(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, 1) - \widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, 0) \right). \tag{4.4}$$

Intuitively, this measure represents the information gain for θ if we observe the difference in potential outcomes $\mathrm{Y}^1 - \mathrm{Y}^0$ for a given measurement $\boldsymbol{x}$ and $\mathcal{D}_{\text{train}}$.

However, a fundamental flaw with this measure exists: observing labels for the random variable $\mathrm{Y}^1 - \mathrm{Y}^0$ is impossible. Thus, τ–BALD represents an irreducible measure of uncertainty. That is, τ–BALD will be high if it is uncertain about the label given the unobserved treatment t', regardless of its certainty about the outcome given the factual treatment t, which makes τ–BALD highest for low-density regions and regions with no overlap. Figure 4.6c 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 have limited overlap, it leads to uninformative acquisitions.

One remedy to the issues of τ–BALD is to only focus on reducible uncertainty:

Definition 4.5.3. *μ–BALD*

$$\mathrm{I}(\mathrm{Y}^{\mathrm{t}}; \theta \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}_{\text{train}}) \approx \underset{\boldsymbol{w} \sim p(\theta \mid \mathcal{D}_{\text{train}})}{\mathrm{Var}} \left(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}) \right). \tag{4.5}$$

This measure represents the information gain for the model parameters θ if we obtain a label for the observed potential outcome Y^{t} given a data point $(\boldsymbol{x}, \mathrm{t})$ and $\mathcal{D}_{\text{train}}$.

μ–BALD only contains observable quantities; however, it does not account for our belief about the counterfactual outcome. As illustrated in Figure 4.6d, 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, the acquisition function would not be optimally data efficient.

4.5.1 Causal–BALD

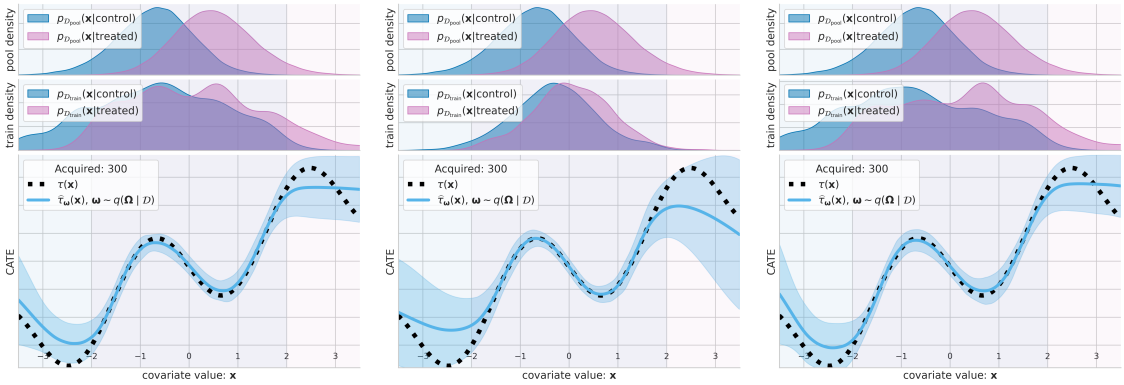
Until now, we looked at simple methods that either considered overlap or considered information gain. Next, we present three measures that account for both factors when choosing a new point to acquire for model training.

A straightforward to combine knowledge about a data point's information gain and overlap is to simply multiply μ–BALD (Equation 4.5) by the propensity acquisition term (Equation 4.3):

Definition 4.5.4. *$\mu\pi$–BALD*

$$\mathrm{I}(\mu\pi \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}_{\text{train}}) \equiv (1 - \widehat{\pi}_{\mathrm{t}}(\boldsymbol{x})) \underset{\boldsymbol{w} \sim p(\theta \mid \mathcal{D}_{\text{train}})}{\mathrm{Var}} \left(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}) \right) \tag{4.6}$$

We can see in Figure 4.7a that the acquisition of training data results in matched sampling that we saw for propensity acquisition in Figure 4.6b. However, the tails of the overlapping distributions extend further into the low-density regions of the pool set support where the overlap assumption is satisfied.



(a) $\mu\pi$–BALD **(b)** ρ–BALD **(c)** $\mu\rho$–BALD

Figure 4.7: Causal–BALD acquisition functions. In all cases, we run active learning for a fixed budget of 300 acquisitions. We see that both $\mu\pi$ and $\mu\rho$–BALD have overlap and obtain a diverse set of acquisitions. ρ–BALD has good overlap, but not diversity. We discuss why this is the case in more detail in Section 4.5.1

Alternatively, we can take an information-theoretic approach to combine knowledge about a data point's information gain and overlap. Let $\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t})$ be an instance of the random variable $\widehat{\mu}^{\mathrm{t}}_{\theta} \in \mathbb{R}$ corresponding to the expected outcome conditioned on t. Further, let $\widehat{\tau}_{\boldsymbol{w}}(\boldsymbol{x})$ be an instance of the random variable $\widehat{\tau}_{\theta} = \widehat{\mu}^{1}_{\theta} - \widehat{\mu}^{0}_{\theta}$ corresponding to the CATE. Then,

Definition 4.5.5. *ρ–BALD*

$$\mathrm{I}(\mathrm{Y}^{\mathrm{t}}; \widehat{\tau}_{\theta} \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}_{\text{train}}) \gtrapprox \frac{1}{2} \log \left(\frac{\mathrm{Var}_{\boldsymbol{w}}\left(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t})\right) - 2\,\mathrm{Cov}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}), \widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}'))}{\mathrm{Var}_{\boldsymbol{w}}\left(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}')\right)} + 1 \right). \tag{4.7}$$

This measure represents the information gain for the CATE τ_{θ} if we observe the outcome Y for a data point ($\boldsymbol{x}$, t) and the data we have trained on $\mathcal{D}_{\text{train}}$.

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 $\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}')$. This scaling biases acquisition towards examples for which we know about the counterfactual outcome, so we can assume that overlap is satisfied for observed $(\boldsymbol{x}, \mathrm{t})$. Second, ρ–BALD is discounted by $\mathrm{Cov}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}), \widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}'))$. This discounting is a concept that we will leave for future discussion.

In Figure 4.7b we see that ρ–BALD has matched the distributions of the treated and control groups similarly to propensity acquisition in Figure 4.6b.

Further, we see that the CATE estimator is more accurate over the support of the data. There is, however, a shortcoming of ρ–BALD that may lead to suboptimal data efficiency. Consider two examples in $\mathcal{D}_{\text{pool}}$, $(\boldsymbol{x}_1, \mathrm{t}_1)$ and $(\boldsymbol{x}_2, \mathrm{t}_2)$ where $\text{Var}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_1, \mathrm{t}_1)) = \text{Var}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_1, \mathrm{t}_1'))$ and $\text{Var}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_2, \mathrm{t}_2)) = \text{Var}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_2, \mathrm{t}_2'))$: for each point, we are as uncertain about the conditional expectation given the factual treatment as we are uncertain given the counterfactual treatment. Further, let $\text{Cov}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_1, \mathrm{t}_1), \widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_1, \mathrm{t}_1')) = \text{Cov}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_2, \mathrm{t}_2), \widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_2, \mathrm{t}_2'))$. Finally, let $\text{Var}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_1, \mathrm{t}_1)) > \text{Var}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}_2, \mathrm{t}_2))$: we are more uncertain about the conditional expectation given the factual treatment for data point $(\boldsymbol{x}_1, \mathrm{t}_1)$ than we are for data point $(\boldsymbol{x}_2, \mathrm{t}_2)$. Under these three conditions, ρ–BALD would rank these two points equally, and so this method would bias training data to the modes of $\mathcal{D}_{\text{pool}}$ when $(\boldsymbol{x}_2, \mathrm{t}_2)$ is more frequent than $(\boldsymbol{x}_1, \mathrm{t}_1)$. In practice, it may be more data-efficient to choose $(\boldsymbol{x}_1, \mathrm{t}_1)$ over $(\boldsymbol{x}_2, \mathrm{t}_2)$ as it would more likely be a point until now unseen by the model.

To combine the positive attributes of μ–BALD and ρ–BALD, while mitigating their shortcomings, we introduce $\mu\rho$–BALD.

Definition 4.5.6. *$\mu\rho$–BALD*

$$\mathrm{I}(\mu\rho \mid \boldsymbol{x}, \mathrm{t}, \mathcal{D}_{\text{train}}) \equiv \underset{\boldsymbol{w}}{\text{Var}}\left(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t})\right) \frac{\text{Var}_{\boldsymbol{w}}(\widehat{\tau}_{\boldsymbol{w}}(\boldsymbol{x}))}{\text{Var}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}'))}. \tag{4.8}$$

Here, we scale Equation 4.7, which has equivalent expression $\frac{\text{Var}_{\boldsymbol{w}}(\widehat{\tau}_{\boldsymbol{w}}(\boldsymbol{x}))}{\text{Var}_{\boldsymbol{w}}(\widehat{\mu}_{\boldsymbol{w}}(\boldsymbol{x}, \mathrm{t}'))}$ by our measure for μ–BALD 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 4.7c that the training data acquisition is distributed more uniformly over the support of the pool data where the overlap assumption is satisfied. Furthermore, the accuracy of the CATE estimator is the highest over that region.

4.5.2 Experimental 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 heterogeneous effect (PEHE), which is defined as $\sqrt{\epsilon_{PEHE}} = \sqrt{\frac{1}{N}\sum_x \left(\hat{\tau}(x) - \tau(x)\right)^2}$. We summarise

Table 4.6: Summary of active learning parameters for each dataset.

Dataset	|Warm-up|	|Acq batch|	Acq steps	|Pool|	|Valid|
Synthetic	10	10	30	10k	1k
IHDP	100	10	38	471	201
CMNIST	250	50	55	35k	15k

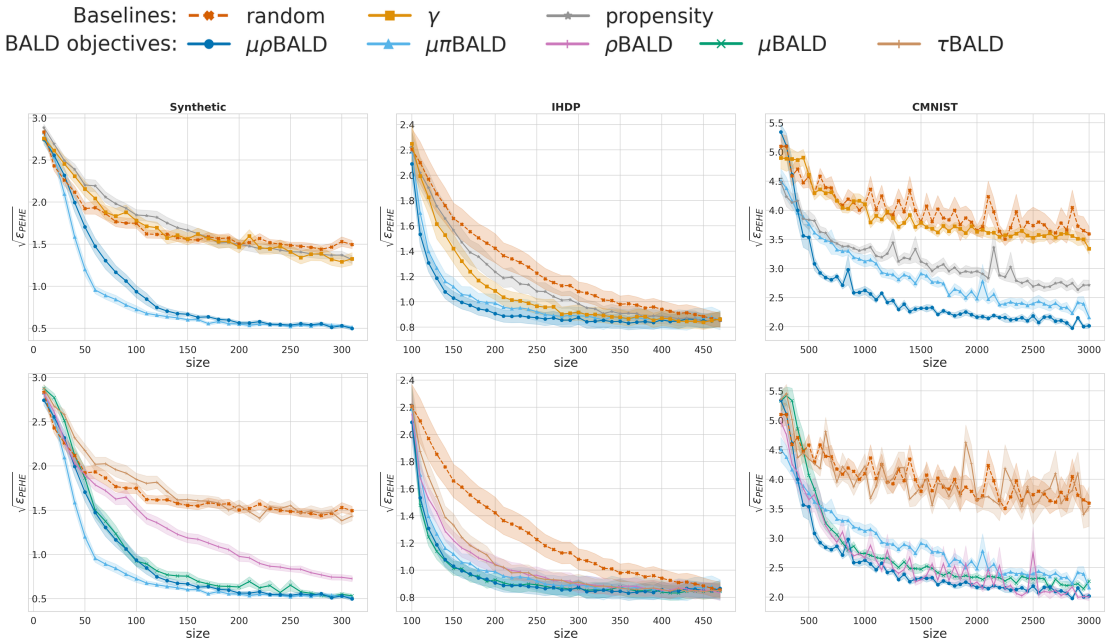


Figure 4.8: $\sqrt{\epsilon_{PEHE}}$ performance (shaded standard error) on IHDP. All results use the DUE model. We observe that BALD objectives outperform the **random**, $\boldsymbol{\gamma}$ and **propensity** acquisition functions significantly, suggesting that epistemic uncertainty aware methods that target reducible uncertainty can be more sample efficient.

each active learning setup in Table 4.6. The **|Warm-up|** is the number of examples in the initial training dataset. **|Acq batch|** is the number of examples labelled at each acquisition step. **Acq steps** is the number of times we query a batch of labels. **|Pool|** is the number of examples in the pool dataset. Finally, **|Valid|** is the number of examples used for model selection when optimizing the model at each acquisition step.

In Figure 4.8, we see that epistemic uncertainty aware $\mu\rho$–BALD outperforms the baselines, random, propensity, and S-Type error rate ($\boldsymbol{\gamma}$). This improvement is expected as the acquisition objective targets reducible uncertainty – that is, epistemic uncertainty when there is 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 4.7c.

Each of (μ–BALD, ρ–BALD, $\mu\pi$–BALD, and $\mu\rho$–BALD) outperforms the baseline methods on these tasks. Of note, the performance ρ–BALD improves as the dimensionality of the covariates increases. In contrast, the performance of the propensity score-based $\mu\pi$–BALD worsens as the dimensionality of the covariates increases. Propensity score estimation is known to be a problem in high-dimensions (D'Amour et al., 2021). We see that both μ-BALD and $\mu\rho$–BALD perform consistently as dimensionality increases, with $\mu\rho$–BALD showing a statistically significant improvement over μ–BALD on two of the three tasks. These improvements indicate that $\mu\rho$–BALD is more robust for data with high-dimensional covariates than $\mu\pi$–BALD; moreover, $\mu\rho$-BALD does not need an additional propensity score model.

4.6 Conclusion and Limitations

We demonstrated that DUE outperforms alternative single forward pass methods on CIFAR-10, and obtains SotA performance in a personalised medicine regression benchmark. We further show that by carefully considering the acquisition function in combination with using DUE, it is viable to do active learning in the context of causal inference. These results show that DUE overcomes the previous problems with uncertainty in DKL, and makes DKL a more reliable method for uncertainty estimation.

DUE meets all desiderata set out in Chapter 1, but there are still a number of limitations worth mentioning. First, the spectral norm coefficient (maximum singular value) in DUE is set to 3, despite needing to be less than 1 to make the model bi-Lipschitz (Behrmann et al., 2019). Additionally, the model employs downsampling, which makes it non Lipschitz in general. These limitations are discussed in Smith et al. (2021), who give insight into why this regularisation method works well despite downsampling using a frequency analysis perspective.

Second, while inference in DUE's output GP is independent of dataset size it is dependent on the number of classes. In particular, on ImageNet size datasets that have over 1,000 classes, inference becomes a bottleneck again. Lastly, implementing

an inducing point GP is significantly more complex than the softmax function. Recent frameworks, such as GPyTorch (Gardner et al., 2018), have made this easier, but it remains a barrier.

5

BatchBALD: Batch Acquisition for Deep Bayesian Active Learning

Contents

This chapter diverges from the topic of single forward pass uncertainty towards principled uncertainty estimation for Active Learning (AL) (Cohn et al., 1996). AL is a powerful technique for attaining data efficiency, a key problem in deep learning. Instead of a-priori collecting and labelling a large dataset, which often comes at

a significant expense, in AL we iteratively acquire labels from an expert only for the most informative data points from a pool of available unlabelled data. After each acquisition step, the newly labelled points are added to the training set, and the model is retrained. This process is repeated until a suitable level of accuracy is achieved, or the labelling budget is spent. The goal of AL is minimise the amount of data that needs to be labelled to achieve that accuracy, or equivalently maximise accuracy within the labelling budget. AL has already made real-world impact in manufacturing (Tong, 2001), robotics (Calinon et al., 2007), recommender systems (Adomavicius et al., 2005), medical imaging (Hoi et al., 2006), and NLP (Siddhant et al., 2018), motivating the need for pushing AL even further.

In AL, the informativeness of new points is assessed by an *acquisition function*. In this chapter, we focus on BALD (Houlsby et al., 2011), which has proven itself in the context of deep learning (Gal et al., 2017; Janz et al., 2017; Shen et al., 2018). BALD is based on mutual information and scores points based on how well their label would inform us about the true model parameter distribution. In deep learning models (Simonyan et al., 2015; He et al., 2016), we generally treat the parameters as point estimates instead of distributions. However, Bayesian neural networks have become a powerful alternative to traditional neural networks and do provide a distribution over their parameters. Improvements in approximate inference (Blundell et al., 2015; Gal et al., 2016; Maddox et al., 2019) have enabled their usage for high dimensional data such as images and in conjunction with BALD for Bayesian AL (Gal et al., 2017).

In practical AL applications, batches of data points are selected during each acquisition step to reduce the number of times the model has to be retrained and expert-time is requested. Model retraining becomes a computational bottleneck for larger models while 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.

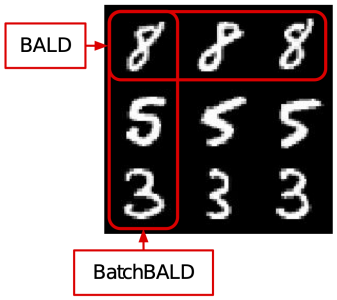


Figure 5.1: Idealised acquisitions of BALD and BatchBALD. If a dataset were to contain many (near) replicas, then BALD would select all replicas of a single informative data point at the expense of other informative data points, wasting data efficiency.

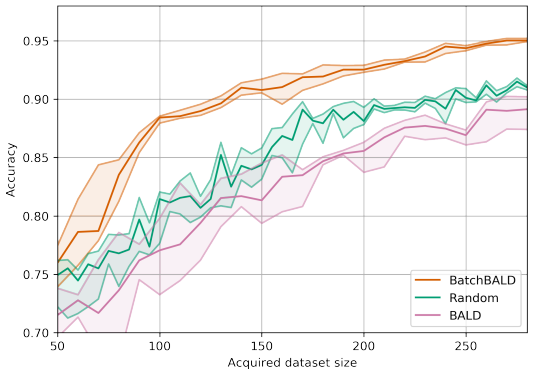


Figure 5.2: Performance on Repeated MNIST with acquisition size 10. See Section 5.3.1 for further details. BatchBALD outperforms BALD while BALD performs worse than random acquisition due to the replications in the dataset.

In Gal et al. (2017), *batch acquisition* is implemented as taking the top b points with the highest BALD acquisition score. This approach leads to acquiring points that are individually very informative, but not necessarily so jointly. See Figure 5.1 for such a batch acquisition of BALD in which it performs poorly whereas scoring points jointly ("BatchBALD") can find *batches* of informative data points. Figure 5.2 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.

Without approximations, finding the best batch to label requires enumerating all possible subsets within the available data, which is intractable as the number of potential subsets grows exponentially with the acquisition size b 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 code for this chapter is publicly available.[1]

[1] `https://github.com/BlackHC/BatchBALD`

The main contributions of this work are:

1. *BatchBALD*, a data-efficient active learning method that selects *sets* of high-dimensional image data to label, leading to improved data efficiency and reduced total run time, Section 5.2;

2. a greedy algorithm to select a batch of points efficiently, Section 5.2.1; and

3. an estimator for the acquisition function that scales to larger acquisition sizes and to datasets with many classes, Section 5.2.2.

5.1 Background

We now review and extend the definition of AL given in Section 2.4. We also revisit MC dropout BNNs (previously discussed in 2.2.2) as we use them to evaluate BatchBALD.

5.1.1 Problem Setting

The Bayesian active learning setup consists of an unlabelled dataset $\mathcal{D}_{\text{pool}}$, the current training set $\mathcal{D}_{\text{train}}$, a Bayesian model $\mathcal{M}$ with model parameters $\boldsymbol{w} \sim \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})$, and output predictions $\mathrm{p}(y \mid \boldsymbol{x}, \theta, \mathcal{D}_{\text{train}})$ for data point $\boldsymbol{x}$ and prediction $y \in \{1, ..., c\}$ in the classification case. The conditioning of $\boldsymbol{w}$ 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 unlabelled 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 $\{\boldsymbol{x}_1^*, ..., \boldsymbol{x}_b^*\}$ is selected using an acquisition function a which scores a candidate batch of unlabelled data points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\} \subseteq \mathcal{D}_{\text{pool}}$ using the current model parameters $\mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})$:

$$\{\boldsymbol{x}_1^*, ..., \boldsymbol{x}_b^*\} = \underset{\{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\} \subseteq \mathcal{D}_{\text{pool}}}{\arg\max} a\left(\{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})\right). \tag{5.1}$$

5.1.2 BALD

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. Intuitively, it captures how strongly the model predictions for a given data point and the model parameters are coupled, implying that finding out about the true label of data points with high mutual information would also inform us about the true model parameters. Originally introduced outside the context of deep learning, the only requirement on the model is that it is Bayesian. BALD is defined as:

$$\mathbb{I}(y\,;\,\theta \mid \boldsymbol{x}, \mathcal{D}_{\text{train}}) = \mathbb{H}(y \mid \boldsymbol{x}, \mathcal{D}_{\text{train}}) - \mathbb{E}_{\mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})}\left[\mathbb{H}(y \mid \boldsymbol{x}, \theta, \mathcal{D}_{\text{train}})\right]. \tag{5.2}$$

Looking at the two terms in Equation 5.2, 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 posterior draws are disagreeing among themselves.

BALD was originally intended for acquiring individual data points and immediately retraining the model. This becomes a bottleneck in deep learning, where retraining takes a substantial amount of time. Applications of BALD (Gal et al., 2016; Janz et al., 2017) usually acquire the top b. This can be expressed as summing over individual scores:

$$a_{\texttt{BALD}}\left(\{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})\right) = \sum_{i=1}^{b} \mathbb{I}(y_i\,;\,\theta \mid \boldsymbol{x}_i, \mathcal{D}_{\text{train}}), \tag{5.3}$$

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

5.1.3 Bayesian Neural Networks (BNN)

In this chapter, we focus on approximate BNNs as our Bayesian model because they scale well to high dimensional inputs, such as images. Compared to regular neural networks, BNNs maintain a distribution over their weights instead of point estimates. Similar to Gal et al. (2017), we use MC dropout (Gal et al., 2016) as a variational approximation for learning a BNN, which is easy to implement, scales well to large models and datasets, and is straightforward to optimise.

5.2 Methods

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:[2]

$$a_{\texttt{BatchBALD}}\left(\{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}, \mathrm{p}(\theta \mid \mathcal{D}_{\mathrm{train}})\right) = \mathbb{I}(y_1, ..., y_b\,;\, \theta \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_b, \mathcal{D}_{\mathrm{train}}). \tag{5.4}$$

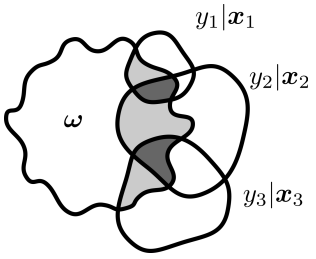
This builds on the insight that independent selection of a batch of data points leads to data inefficiency as correlations between data points in an acquisition batch are not taken into account.

To understand how to compute the mutual information between a set of points and the model parameters, we express $\boldsymbol{x}_1, ..., \boldsymbol{x}_b$, and $y_1, ..., y_b$ through joint random variables $\boldsymbol{x}_{1:b}$ and $y_{1:b}$ in a product probability space and use the definition of the mutual information for two random variables:

$$\mathbb{I}(y_{1:b}\,;\, \theta \,|\, \boldsymbol{x}_{1:b}, \mathcal{D}_{\mathrm{train}}) = \mathbb{H}(y_{1:b} \,|\, \boldsymbol{x}_{1:b}, \mathcal{D}_{\mathrm{train}}) - \mathbb{E}_{\mathrm{p}(\theta|\mathcal{D}_{\mathrm{train}})}\, \mathbb{H}(y_{1:b} \mid \boldsymbol{x}_{1:b}, \theta, \mathcal{D}_{\mathrm{train}}). \tag{5.5}$$

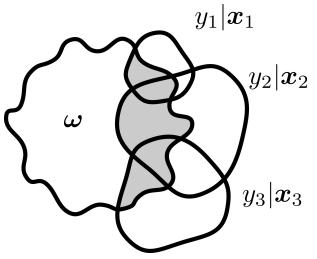
Intuitively, the mutual information between two random variables can be seen as the intersection of their information content. In fact, Yeung (1991) shows that a signed measure μ^* can be defined for discrete random variables x, y, such that $\mathbb{I}(x\,;\,y) = \mu^*(x \cap y)$, $\mathbb{H}(x, y) = \mu^*(x \cup y)$, $\mathbb{E}_{\mathrm{p}(y)}\, \mathbb{H}(x \mid y) = \mu^*(x \setminus y)$, and so on,

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



$$\sum_i \mathbb{I}(y_i\,;\,\theta \mid \boldsymbol{x}_i, \mathcal{D}_{\text{train}})$$

(a) BALD



$$\mathbb{I}(y_1, ..., y_b\,;\,\theta \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_b, \mathcal{D}_{\text{train}})$$

(b) BatchBALD

Figure 5.3: 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 θ. Areas contributing to the respective score are shown in grey, and areas that are double-counted in dark grey.

where we identify random variables with their counterparts in information space, and conveniently drop conditioning on $\mathcal{D}_{\text{train}}$ and $\boldsymbol{x}_i$.

Using this, BALD can be viewed as the sum of individual intersections:

$$\sum_i \mu^*(y_i \cap \theta), \tag{5.6}$$

which double counts overlaps between the y_i. Extending BALD to the mutual information between $y_1, ..., y_b$ and θ, which is equivalent to $\mu^*(\bigcap_i y_i \cap \theta)$, would lead to selecting *similar* data points instead of diverse ones under maximisation.

BatchBALD takes overlaps into account by computing $\mu^*(\bigcup_i y_i \cap \theta)$ and is more likely to acquire a more diverse cover under maximisation:

$$\mathbb{I}(y_{1:b}\,;\,\theta \mid \boldsymbol{x}_{1:b}\mathcal{D}_{\text{train}}) \tag{5.7}$$

$$= \mathbb{H}(y_{1:b} \mid \boldsymbol{x}_{1:b}, \mathcal{D}_{\text{train}}) - \mathbb{E}_{\mathrm{p}(\theta|\mathcal{D}_{\text{train}})}\, \mathbb{H}(y_{1:b} \mid \boldsymbol{x}_{1:b}, \theta, \mathcal{D}_{\text{train}}) \tag{5.8}$$

$$= \mu^*\left(\bigcup_i y_i\right) - \mu^*\left(\bigcup_i y_i \setminus \theta\right) = \mu^*\left(\bigcup_i y_i \cap \theta\right) \tag{5.9}$$

This is depicted in Figure 5.3 and also motivates that $a_{\texttt{BatchBALD}} \leq a_{\texttt{BALD}}$. We can prove this by using the subadditivity property of information entropy and the independence of the y_i given θ, we show that BALD is an approximation of

Algorithm 3 Greedy BatchBALD $1 - 1/e$-approximate algorithm

Input: acquisition size b, unlabelled dataset $\mathcal{D}_{\text{pool}}$, model parameters $\mathrm{p}(\theta|\mathcal{D}_{\text{train}})$
$A_0 \leftarrow \emptyset$
for $n \leftarrow 1...b$ do **do**
  for $\boldsymbol{x} \in \mathcal{D}_{\text{pool}} \setminus A_{n-1}$ **do**
    $s_{\boldsymbol{x}} \leftarrow a_{\texttt{BatchBALD}}\left(A_{n-1} \cup \{\boldsymbol{x}\}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})\right)$
  end for
  $\boldsymbol{x}_n \leftarrow \underset{\boldsymbol{x} \in \mathcal{D}_{\text{pool}} \setminus A_{n-1}}{\arg\max} \; s_{\boldsymbol{x}}$
  $A_n \leftarrow A_{n-1} \cup \{\boldsymbol{x}_n\}$
end for
Output: acquisition batch $A_n = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}$

BatchBALD and is always an upper bound on the respective BatchBALD score:

$$a_{\texttt{BatchBALD}}\left(\{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})\right) \tag{5.10}$$

$$= \mathbb{H}(y_1, ..., y_b \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_b, \mathcal{D}_{\text{train}}) - \mathbb{E}_{\mathrm{p}(\theta|\mathcal{D}_{\text{train}})}\left[\mathbb{H}(y_1, ..., y_b \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_b, \theta, \mathcal{D}_{\text{train}})\right] \tag{5.11}$$

$$\leq \sum_{i=1}^{b} \mathbb{H}(y_i \mid \boldsymbol{x}_i, \mathcal{D}_{\text{train}}) - \sum_{i=1}^{b} \mathbb{E}_{\mathrm{p}(\theta|\mathcal{D}_{\text{train}})}\left[\mathbb{H}(y_i \mid \boldsymbol{x}_i, \theta, \mathcal{D}_{\text{train}})\right] \tag{5.12}$$

$$= \sum_{i=1}^{b} \mathbb{I}(y_i\,;\, \theta \mid \boldsymbol{x}_i, \mathcal{D}_{\text{train}}) = a_{\texttt{BALD}}\left(\{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})\right) \tag{5.13}$$

For acquisition size 1, BatchBALD and BALD are equivalent, which we prove in Appendix C.3.

5.2.1 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 3. In Appendix C.2, we prove that $a_{\texttt{BatchBALD}}$ is submodular, which means the greedy algorithm is $1 - 1/e$-approximate (Nemhauser et al., 1978; Krause et al., 2008; Cuong et al., 2013).

In Appendix C.3, we show that, under idealised conditions, when using BatchBALD and a fixed final $|\mathcal{D}_{\text{train}}|$, the active learning loop itself can be seen as a greedy $1 - 1/e$-approximation algorithm, and that an active learning loop with BatchBALD and acquisition size larger than 1 is bounded by an active learning

loop with individual acquisitions, that is BALD/BatchBALD with acquisition size 1, which is the ideal case.

5.2.2 Computing BatchBALD

For brevity, we leave out conditioning on $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$, and $\mathcal{D}_{\text{train}}$, and $\mathrm{p}(\theta)$ denotes $\mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})$ in this section. $a_{\texttt{BatchBALD}}$ is then written as:

$$a_{\texttt{BatchBALD}}\left(\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}, \mathrm{p}(\theta)\right) = \mathbb{H}(y_1, ..., y_n) - \mathbb{E}_{\mathrm{p}(\theta)}\left[\mathbb{H}(y_1, ..., y_n \mid \theta)\right]. \tag{5.14}$$

Because the y_i are independent when conditioned on θ, computing the right term of Equation 5.14 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 $\hat{\boldsymbol{w}}_j \sim \mathrm{p}(\theta)$:

$$\mathbb{E}_{\mathrm{p}(\theta)}\left[\mathbb{H}(y_1, ..., y_n \mid \theta)\right] = \sum_{i=1}^{n} \mathbb{E}_{\mathrm{p}(\theta)}\left[\mathbb{H}(y_i \mid \theta)\right] \approx \frac{1}{k} \sum_{i=1}^{n} \sum_{j=1}^{k} \mathbb{H}(y_i \mid \hat{\boldsymbol{w}}_j). \tag{5.15}$$

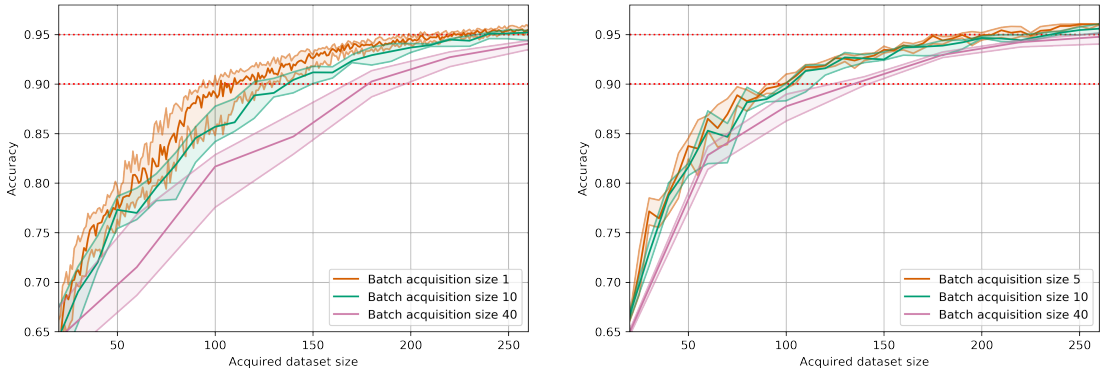
Computing the left term of Equation 5.14 is difficult because the unconditioned joint probability does not factorise. Applying the equality $\mathrm{p}(y) = \mathbb{E}_{\mathrm{p}(\theta)}\left[\mathrm{p}(y \mid \theta)\right]$, and, using sampled $\hat{\boldsymbol{w}}_j$, we compute the entropy by summing over all possible configurations $\hat{y}_{1:n}$ of $y_{1:n}$:

$$\begin{aligned}
\mathbb{H}(y_1, ..., y_n) &= \mathbb{E}_{\mathrm{p}(y_1, ..., y_n)}\left[-\log \mathrm{p}(y_1, ..., y_n)\right] && (5.16)\\
&= \mathbb{E}_{\mathrm{p}(\theta)}\, \mathbb{E}_{\mathrm{p}(y_1, ..., y_n \mid \theta)}\left[-\log \mathbb{E}_{\mathrm{p}(\theta)}\left[\mathrm{p}(y_1, ..., y_n \mid \theta)\right]\right] && (5.17)\\
&\approx -\sum_{\hat{y}_{1:n}} \left(\frac{1}{k} \sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{w}}_j)\right) \log \left(\frac{1}{k} \sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{w}}_j)\right). && (5.18)
\end{aligned}$$

5.2.3 Efficient Implementation

In each iteration of the algorithm, $\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}$ stay fixed while $\boldsymbol{x}_n$ varies over $\mathcal{D}_{\text{pool}} \setminus A_{n-1}$. We can reduce the required computations by factorizing $\mathrm{p}(y_{1:n} \mid \theta)$ into $\mathrm{p}(y_{1:n-1} \mid \theta)\, \mathrm{p}(y_n \mid \theta)$. We store $\mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{w}}_j)$ in a matrix $\hat{P}_{1:n-1}$ of shape $c^{n-1} \times k$ and $\mathrm{p}(y_n \mid \hat{\boldsymbol{w}}_j)$ in a matrix $\hat{P}_n$ of shape $c \times k$. The sum $\sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{w}}_j)$ in Equation 5.18 can then be turned into a matrix product:

$$\frac{1}{k} \sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{w}}_j) = \frac{1}{k} \sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{w}}_j)\, \mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{w}}_j) = \left(\frac{1}{k} \hat{P}_{1:n-1} \hat{P}_n^T\right)_{\hat{y}_{1:n-1}, \hat{y}_n}. \tag{5.19}$$



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

Figure 5.4: Classification performance on MNIST for several acquisition sizes. BALD's performs significantly worse for larger acquisition sizes, because it does not consider the joint informativeness and acquires repeated points in each batch. Meanwhile, BatchBALD maintains strong performance even at larger acquisition size. However, we do see performance degradation at 40 points, which is due to the noise in the approximations required to compute BatchBALD at that size.

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 $\mathrm{p}(y \mid \hat{\boldsymbol{w}}_j)$ 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 C.4 for details.

Monte-Carlo sampling bounds the time complexity of the full BatchBALD algorithm to $\mathcal{O}(bc \cdot \min\{c^b, m\} \cdot |\mathcal{D}_{\text{pool}}| \cdot k)$ compared to $\mathcal{O}(c^b \cdot |\mathcal{D}_{\text{pool}}|^b \cdot k)$ for finding the exact optimal batch and $\mathcal{O}((b+k) \cdot |\mathcal{D}_{\text{pool}}|)$ for BALD (remind that b is the acquisition size, c is the number of classes, k is the number of MC dropout samples).

5.3 Experiments

We start by showing how an application of the BALD algorithm in the batch setting 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 favourable results (Figure 5.2).

We then illustrate BatchBALD's effectiveness on standard AL datasets: MNIST and EMNIST. EMNIST (Cohen et al., 2017) is an extension of MNIST that also

Figure 5.5: Examples of all 47 classes of EMNIST.

includes letters, for a total of 47 classes, and has a training set that is double the size of MNIST's. See Figure 5.5 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 finetune a pre-trained model on a more difficult dataset called CINIC-10 (Darlow et al., 2018), which is a combination of CIFAR-10 and downscaled ImageNet.

Experimental Setup In our experiments, we repeatedly go through active learning loops. One active learning loop consists of training the model on the available labelled data and subsequently acquiring new data points using a chosen acquisition function. As the labelled 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 optimiser (Kingma et al., 2014a) with learning rate 0.001, β_1 0.9, and β_2 0.999 (the default values in PyTorch (Paszke et al., 2019)). 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 the figures in this chapter.

We reinitialise the model after each acquisition, similar to Gal et al. (2017), and found this helps the model improve even when few data points are acquired. It

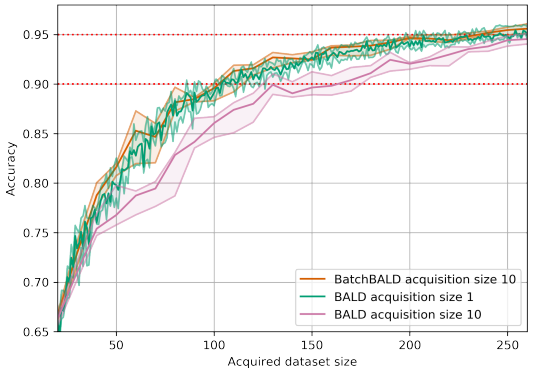


Figure 5.6: Accuracy on MNIST for BALD and BatchBALD. BatchBALD outperforms BALD with acquisition size 10 and performs close to the optimum of BALD with acquisition size 1.

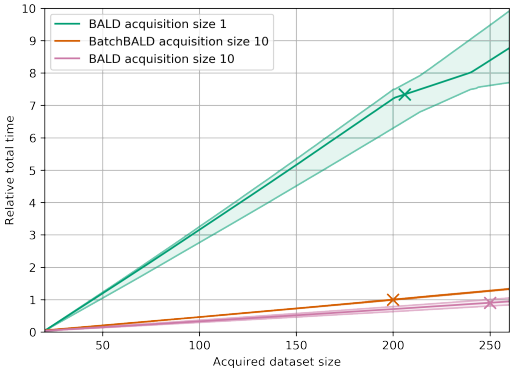


Figure 5.7: Relative total time on MNIST. Normalised to training BatchBALD with acquisition size 10 to 95% accuracy. The stars mark when 95% accuracy is reached for each method.

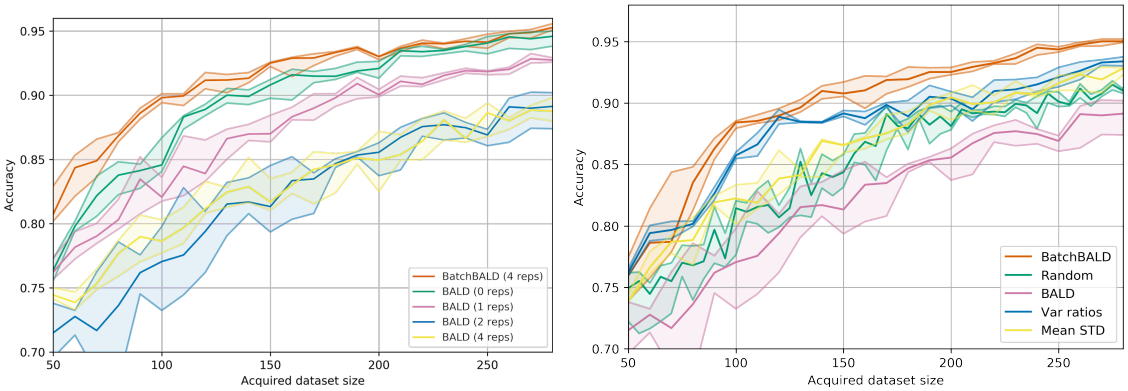
also decorrelates subsequent acquisitions as final model performance is dependent on a particular initialisation (Frankle et al., 2019).

MC Dropout When computing $\mathrm{p}(\boldsymbol{y} \mid \boldsymbol{x}, \theta, \mathcal{D}_{\text{train}})$, it is important to use the same dropout masks for each data point $\boldsymbol{x}$ when sampling from the model with MC dropout. 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. Note that the default implementation in most machine learning frameworks uses a *different* mask per element of a minibatch. 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 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.

5.3.1 Repeated MNIST

As demonstrated in the introduction, applying BALD to a dataset that contains many (near) replicated data points leads to poor performance. We show how this



(a) Increasing Repetitions **(b)** Alternative Method Comparison

Figure 5.8: Performance on Repeated MNIST. In Figure 5.8a, we see that BALD performs worse as the number of repetitions is increased, while BatchBALD outperforms BALD with zero repetitions. In Figure 5.8b, we compare BALD, BatchBALD, Var Ratios (Freeman, 1965), Mean STD (Kendall et al., 2015) and random acquisition with acquisition size 10 and 10 MC dropout samples.

manifests in practice by taking the MNIST dataset and replicating each data point in the training set two times (obtaining a training set that is three times larger than the original). After normalising 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, two of each class (similar to Gal et al. (2017)). These initial data points were chosen by running an active learning loop with BALD and an initial dataset picked randomly 6 times and choosing the set of the median model. They were subsequently held fixed. The model architecture used for this dataset is described in Section C.1.

The results can be seen in Figure 5.2. In this illustrative scenario, BALD performs poorly, and even randomly acquiring points performs better. However, BatchBALD is able to cope with the replication perfectly.

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 5.8a. BatchBALD performs the same on all repetition numbers (100 data points till 90%). BALD achieves 90% accuracy at

Table 5.1: 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.

	90% accuracy	95% accuracy
BatchBALD	70 / 90 / 110	190 / 200 / 230
BALD (follows Section 5.3)	120 / 120 / 170	250 / 250 / >300
BALD (Gal et al., 2017)	145	335

120 data points (0 repetitions), 160 data points (1 repetition), 280 data points (2 repetitions), 300 data points (4 repetitions). In Figure 5.8b, we evaluate the other methods used in Gal et al. (2017) on RMNIST. We see that some of these methods are better able to cope with the duplication, such as Var Ratios. We hypothesise that this is due to these methods being more noisy, given that acquiring repeated points *is* the optimal behaviour under summation of informativeness.

5.3.2 MNIST

For the second experiment, we follow the setup of Gal et al. (2017) and perform AL on the MNIST dataset using 100 MC dropout samples. We use the same model architecture and initial dataset as in Section 5.3.1. 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 5.1.

We first look at BALD for increasing acquisition size in Figure 5.4a. 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 5.4b, 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 5.6. 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 visualised in Figure 5.7, where

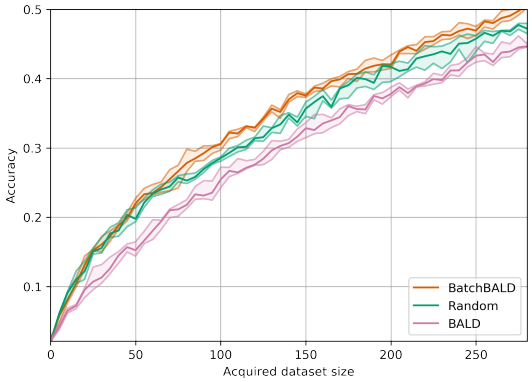


Figure 5.9: Performance on EMNIST. BatchBALD consistently outperforms both random acquisition and BALD while BALD is unable to beat random acquisition.

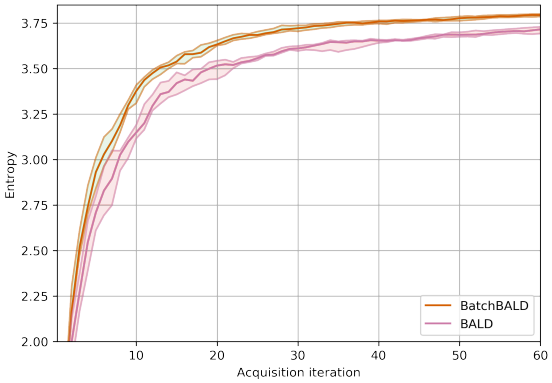


Figure 5.10: Entropy of acquired class labels over acquisition steps on EMNIST. BatchBALD steadily acquires a more diverse set of data points.

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.

5.3.3 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 instead perform the initial acquisition step with the randomly initialised model. We use 10 MC dropout samples for estimating BALD and increase model capacity (see Section C.1 for further details).

The results for acquisition size 5 can be seen in Figure 5.9. BatchBALD outperforms both random acquisition and BALD while BALD is unable to beat random acquisition. Figure 5.10 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 5.11, 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.

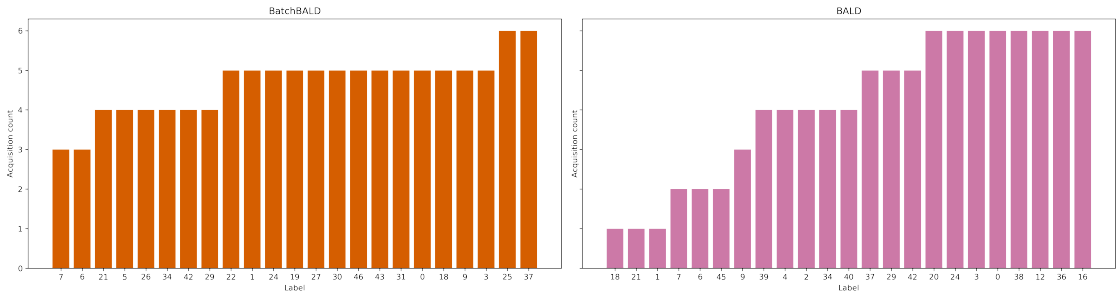


Figure 5.11: Histogram of acquired class labels on EMNIST. BatchBALD left and BALD right. Classes are sorted by number of acquisitions only the lower half of the classes are shown. Several EMNIST classes are underrepresented in BALD and random acquisition while BatchBALD acquires classes more uniformly. The histograms were created from all acquired points at the end of an active learning loop.

5.3.4 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 on the test set it is important to obtain data from both sources. 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 an ImageNet pre-trained model and acquire data only to finetune the original model. This is suitable in cases where there is a large amount of data available in an auxiliary domain, but it is expensive to label for the domain of interest.

For the CINIC-10 experiment, we use 160k training samples for the unlabelled pool, 20k validation samples, and the other 90k as test samples. We use an ImageNet pre-trained VGG-16, provided by PyTorch (Paszke et al., 2019), 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 5.12, with the 59% mark reached at 1170 for BatchBALD and 1330 for BALD (median).

5.4 Related Work

AL is closely related to Bayesian Optimisation (BO), which is concerned with finding the global optimum of a function (Snoek et al., 2012), with the fewest

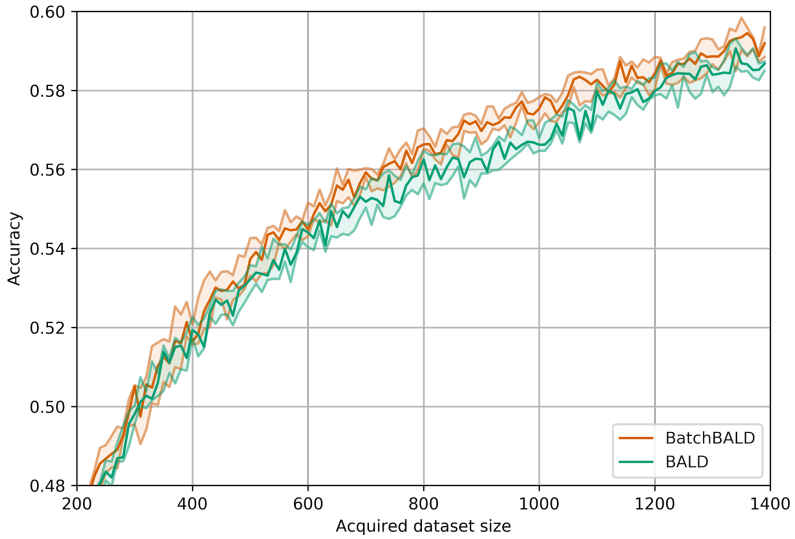


Figure 5.12: Performance on CINIC-10. BatchBALD outperforms BALD from 500 acquired samples onwards.

number of function evaluations. This is generally done using a Gaussian Process. A common problem in BO is the lack of parallelism, with usually a single worker being responsible for function evaluations. In real-world settings, there are usually many such workers available and making optimal use of them is an open problem (González et al., 2016; Alvi et al., 2019) with some work exploring mutual information for optimising a multi-objective problem (Hernández-Lobato et al., 2016).

Maintaining diversity when acquiring a batch of data has also been attempted using constrained optimisation (Guo et al., 2008) and in Gaussian Mixture Models (Azimi et al., 2012). In AL 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.

A related approach to AL is semi-supervised learning (also sometimes referred to as weakly-supervised), in which the labelled data is commonly assumed to be fixed, and the unlabelled data is used for unsupervised learning (Kingma et al., 2014b; Rasmus et al., 2015). Wang et al. (2017), Sener et al. (2018), and Samarth Sinha (2019) explore combining it with AL.

5.5 Scope and Limitations

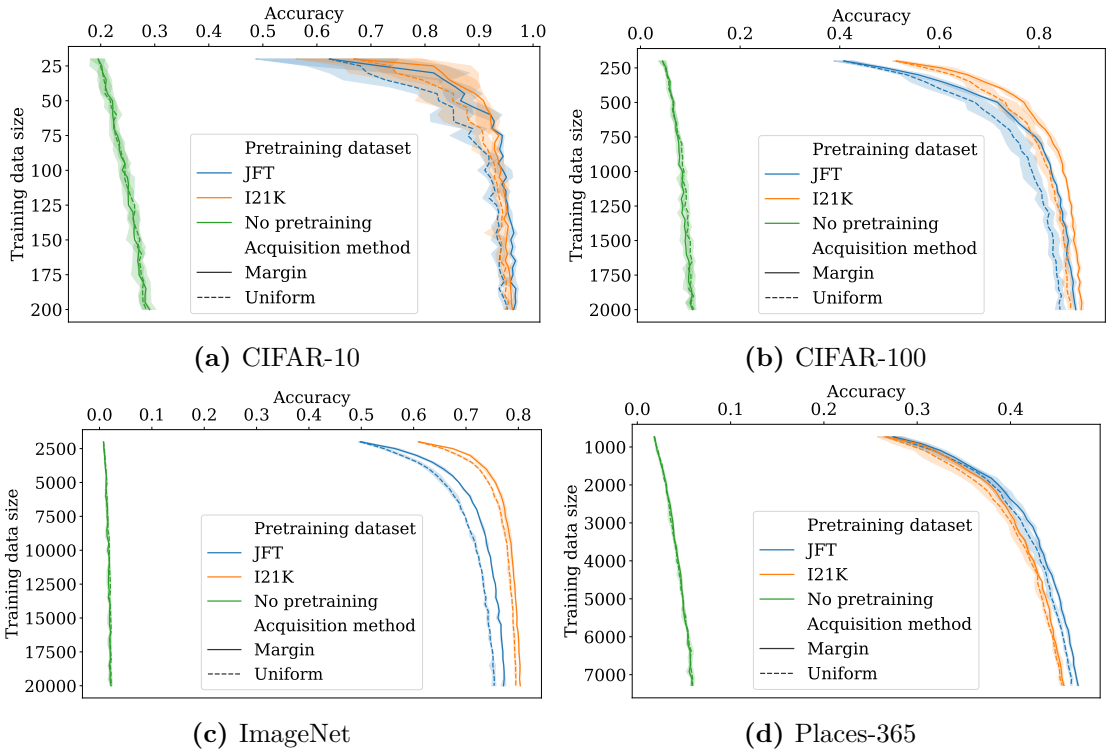
Unbalanced Datasets BALD and BatchBALD do not work well when the test set is unbalanced as they aim to learn well about all classes and do not follow the density of the dataset. However, if the test set is balanced, but the training set is not, we do expect BatchBALD to perform well. Alternatively, one could consider upweighting the scores by the empirical class balance in the test set and focus on acquiring data that is more represented in the test set.

Unlabelled Data BatchBALD does not take into account any information from the unlabelled dataset. However, semi-supervised learning could improve the accuracy and uncertainty estimates of our BNN. 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.

5.6 Active Learning with Very Large Pre-trained Models

We now shift gears and look at active learning with very large pre-trained models. This section is based on work that took place after publishing the previous part of this chapter. Previously pre-trained experiments in this chapter were done using a VGG-19 based model pre-trained on ImageNet. In this section, we propose to go far beyond that by using a ViT-32L (Dosovitskiy et al., 2021) trained on the JFT-4B dataset, which is an internal dataset by Google. Given a model that is trained on a large dataset relevant to the active learning dataset, we can expect the model to obtain excellent performance with just a few labelled examples. Due to large pre-trained models being significantly more general and robust than previous



(a) CIFAR-10 **(b)** CIFAR-100

(c) ImageNet **(d)** Places-365

Figure 5.13: Active learning on (a) CIFAR-10, (b) CIFAR-100, (c) ImageNet and (d) Places365. JFT implies pre-training on the JFT-4B internal Google dataset, I21K is the ImageNet derived dataset with 21,000 classes (Deng et al., 2009). I21K does not share classes with standard ImageNet, but it is more similar to it than JFT. The margin acquisition function (Scheffer et al., 2001) is discussed in Section 2.4. The uniform acquisition function selects data points uniformly at random. There are very significant benefits to pre-training compared to training from scratch in all four scenarios.

generation models (Radford et al., 2021), the chances that the model can transfer to the AL dataset is substantial.

We augment the ViT-L32 with Batch-Ensemble layers (Wen et al., 2020) and pre-train on JFT-4B or ImageNet21K (ImageNet with all 21,000 classes) (Deng et al., 2009). This code for this section is available.[3] In Figure 5.13, we see that these very large pre-trained models significantly outperform their not pre-trained counterparts. On ImageNet, pre-training on the smaller but more relevant dataset Imagenet21K significantly outperforms pre-training on the larger but less relevant dataset JFT-4B. We discuss the importance of the pre-trained dataset further in Section 6.3.3.

[3] `https://github.com/google/uncertainty-baselines`

5.7 Conclusion

We have introduced a new batch acquisition function, BatchBALD, for Deep Bayesian Active Learning, and an approximate 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 labelled 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.

6

Discussion and Open Questions

Contents

In this thesis, we described new methods and insights for uncertainty estimation in deep learning. We answered the research questions set out in Chapter 1, and in the course of doing so several new questions have opened up. First, we look at open questions directly related to the chapters in this thesis. Subsequently, we consider recent advances in very large pre-trained models such as CLIP (Radford et al., 2021) and GPT-3 (Brown et al., 2020) and analyse how the mechanisms devised in this thesis apply to them. We suggest exciting new research directions enabled by the very large pre-trained models and end by summarising the contributions of this thesis and looking back at the most important lessons learned.

6.1 Future of Uncertainty Estimation Evaluation

An important part of introducing new uncertainty estimation techniques is to perform a fair comparison to current methods. In this section, we specifically look at the OOD detection evaluation method, often used to evaluate desideratum 2 of Section 1.1. The goal of this evaluation method is to distinguish between two datasets based on uncertainty (see Section 2.3.3). It is commonly used to evaluate uncertainty estimation techniques in the literature (Lakshminarayanan et al., 2017; Fort et al., 2019; Nalisnick et al., 2019a; Ren et al., 2019; Liu et al., 2020; Ren et al., 2021), and we use it as part of our evaluations in Chapters 3 and 4.

The OOD detection evaluation method corresponds to the deployment scenario where an unexpected and unclassifiable input is encountered, and it needs to be detected. For example, consider an X-ray image taken to detect a broken finger but containing only an elbow. It is not known upfront that the input is unclassifiable, nor is it known during training time what unexpected inputs can be encountered. To be realistic, the evaluation needs to adhere to the following rules:

1. The OOD dataset cannot be used for training or model selection. It is unclear if a model trained with the OOD dataset would generalise to different OOD data and giving access does improve the model significantly (Band et al., 2021).

2. Elements of the OOD dataset are preprocessed in the same way as the original dataset. By preprocessing, we mean the input standardisation done before evaluating the model. This might seem trivial, but in practice the detection task becomes much simpler if this is ignored and given that we do not know at deployment time what data will be encountered, this is the only correct thing to do.

3. Each point should be treated independently. Access to two independent OOD samples at the same time makes the task much simpler (see, e.g., Nalisnick et al. (2019c)), but given that OOD samples are expected to be relatively rare, this is not a reasonable assumption to make.

In the ideal case, the model is evaluated on *several* OOD datasets, and if hyperparameter tuning or model selection is necessary then a completely separate dataset should be used for this.

This set of rules makes the evaluation cumbersome, prone to mistakes, and easy to manipulate. For example, by incorrectly preprocessing the OOD dataset one can get better results, which means this bug is tough to spot. The exact evaluation set-up is often obscure and a cause for discussions during peer review, while manipulations inhibit measuring progress. We now suggest a potential alternative evaluation method which avoids having to consider the rules above and is less prone to mistakes or manipulation. We propose to call this "unclassifiability detection" as an alternative for the unfortunate naming "OOD detection", which, as we discussed in Section 2.3.3 could mean many different types of tasks.

6.1.1 Evaluation of Unclassifiability Detection

We propose an alternative way of evaluating models trained on a given popular dataset (e.g., CIFAR-10). We start by introducing two new evaluation datasets that are disjoint. Each set consists of actual images from the original test set (or equivalently obtained alternatives, see Recht et al. (2018, 2019)), in combination with other images with classes that are not represented in the training set (e.g., taken from ImageNet). The first evaluation set can be used for hyperparameter tuning and model selection (the "validation set") and the second one is used for method comparison only (the "test set"). To reduce potential manipulation, the evaluation of the predictions on the test set can be done by a third party, such as Kaggle. One can compute the AUROC metric of distinguishing between two datasets by post-hoc filtering out predictions on the data with classes not represented in the training data. The AUROC in the selective prediction setting can be computed by considering the full set of predictions. See Section 2.3.3 for more context on these evaluation metrics. Different levels of difficulty could be provided by mixing in different types of datasets and using different ratios of original and added data. Additionally, one can consider multiple different validation and test sets, to allow

evaluating the generalisation properties of proposed methods. Both extensions will also reduce bias arising from the exact data that was mixed in.

The main improvement comes from providing a test set that already includes data from another dataset, instead of considering it separately. This makes the setup similar to standard classification, which means that one must preprocess the individual points in the same way, and that it is not possible to use OOD detection methods that depend on using multiple OOD samples. It is much easier to adhere to the rules laid out in the previous section. This evaluation method is partially inspired by the CINIC-10 dataset (Darlow et al., 2018), which extends CIFAR-10 with images from ImageNet that contain classes that match CIFAR-10. However, that dataset is aimed at evaluating generalisation and not uncertainty estimation.

6.2 Future of Active Learning

In this section, we propose another new evaluation protocol, this time for active learning. The current evaluation protocol of AL gives researchers the maximum amount of freedom, but is unrealistic from a practitioners' perspective. We further discuss a new batch active learning method with DKL based on insights from Chapters 4 and 5. The new method is simpler to implement and faster to compute than BatchBALD.

6.2.1 Evaluating Active Learning

A common issue that researchers in the field of AL face is learning a large model on a small dataset. The small dataset leads to an increased risk of overfitting and often a different set of hyperparameters is required than when training on a large dataset. A robust way to avoid overfitting is to use early-stopping, where training is stopped when the validation loss starts to decrease. Hyperparameter tuning can be done by running the full active learning loop several times and comparing the performance on a separate validation set at the end. These approaches are taken in Chapter 5. Alternatively, one can also change the hyperparameters after each acquisition step, but this will introduce variance in the evaluation and is rarely done

in research. Both these practices run into a fundamental problem when bridging the gap from active learning research to practice: there are no (labelled) validation sets available in practice, just like there is no labelled training set. In addition, it is not possible to run the full active learning loop several times as the labelling budget is spent after the first time. This means that models that are sensitive to their hyperparameter settings are at a large disadvantage in the AL setting. We argue that research in active learning should consider the total amount of labels to successfully train and evaluate a model and only run active learning *once* on their dataset of interest. Adhering to these requirements will need a significant shift from the current approach as researchers will need to consider the sample efficiency of their model evaluation (Kossen et al., 2021). Only evaluating an AL loop on the dataset once is more straightforward: one can use a second dataset for the hyperparameter tuning.

The active learning protocol commonly used in academic research is also *more* difficult than necessary. The model is trained from scratch on a small labelled dataset and the unlabelled dataset is ignored for the purpose of obtaining a trained model. This is an odd choice, given that there are unsupervised training algorithms available for many types of data (Chen et al., 2020), and in some cases there might even be strong relevant pre-trained models available (Tan et al., 2019; Radford et al., 2021). A potential way to make use of these advances is to consider active learning in the transfer learning setting. This would avoid the problems with learning on a small dataset, and greatly reduce the number of hyperparameters that need to be tuned. We looked at the pre-trained setting briefly in Section 5.6 and discuss its limitations in more depth in Section 6.3.3.

6.2.2 Scalable Batch Active Learning

BatchBALD (Chapter 5) introduced the first practical method for computing the informativeness of a batch of data points jointly. It does so by using a greedy approximation of the joint entropy with a bounded error. As mentioned in the limitations in Chapter 5, noise in the estimator of the posterior and further in the joint entropy computation prohibits scaling BatchBALD to batches larger

than 30 to 40. Meanwhile, a GP with a Gaussian likelihood has a closed form solution for computing joint uncertainty on a few thousand points, after which numerical precision makes the Cholesky decomposition difficult. From Chapter 4, we also know that a GP in the context of DKL can obtain excellent uncertainty estimation on large scale datasets.

There are two options for enabling joint uncertainty computation for classification GPs. The first is to compute the uncertainty pre softmax integration, on $q(\boldsymbol{F}^* \mid \boldsymbol{Y}; \boldsymbol{X}, \boldsymbol{X}^*)$ of Section 2.2.4, similar to Kendall et al. (2017). The problem with this approach is that the model could predict a very large mean *and* a large variance for a particular class and a small mean with large variance for the other classes (note that the classes are modelled independently). Pre softmax, the large variance would indicate high uncertainty, but post softmax the large mean value of that particular class would dominate every MC softmax sample, leading to a low entropy predictive softmax distribution and therefore low uncertainty. In effect, the magnitude of the variance needs to scale with the mean value to have meaningful influence on the uncertainty post softmax, but the likelihood pushes the mean values as large as possible as it continues to reduce the likelihood component of the ELBO.

The second option is to avoid the softmax likelihood altogether and turn the classification problem into regression. This can be done by transforming the labels into a one-hot encoding and directly regressing to them. Usually models trained in this way underperform their classification counterpart, but Milios et al. (2018) show that by changing the likelihood to a Dirichlet, which involves smoothing of the one hot encoded label, accuracy and uncertainty quality can match that of a model trained with the softmax likelihood. With this tool in hand, it is possible to evaluate the joint uncertainty in closed form, and do batch Bayesian active learning with DKL. In a preliminary experiment, we were able to confirm Milios et al. (2018)'s finding and obtained an accuracy of 95.7% on CIFAR-10 when training DUE with a Dirichlet observation model, which is a slight improvement over the 95.6% reported for DUE with a Softmax likelihood in Table 4.2. This

underscores the viability of this approach, although more work is necessary to evaluate the quality of the uncertainty estimation.

6.3 Very Large Pre-Trained Models

Deep learning research has long been focused on training deeper models with increasing number of parameters. The hope is that these models will be able to learn more complex behaviour and make more accurate predictions. The field started with a handful of layers (Hinton et al., 2006), then tens of layers (Simonyan et al., 2015), hundreds of layers (He et al., 2015), and now transformers with billions of parameters (Vaswani et al., 2017; Dosovitskiy et al., 2021). Aside from the architectural improvements, there have also been improvements in hardware, software, and larger datasets. These improvements combined have led to models with excellent classification, robustness, and generalisation ability (Radford et al., 2021). Training these very large models is infeasible for most individuals and requires a dedicated team with a large budget. Fortunately, the companies that train these models frequently make them publicly available (Tan et al., 2019; Radford et al., 2021). However, they do not provide the dataset used for training and only describe the process by which it was gathered with minimal details, leaving out important information on data sources, filtering, and total size. This has lead to the situation where both the model, by virtue of being a very large neural network, and the dataset have become black boxes (Breiman, 2001).

Many of the analyses done in this thesis require knowing what data the model is (and is not) trained on. By removing that information, it becomes very difficult to assess how the model will behave on data unlike its training data. Aside from changing the output, architectural changes are not viable either, because the computational cost and private data make it impossible to train a new model. However, given the large performance improvements this type of model brings compared to smaller alternatives, it seems evident that they will become the standard method of deep learning, so new tools for estimating uncertainty and

analysing the model's abilities that can work within these constraints will have to be researched. We discuss potential ways to do so below.

6.3.1 Evaluating Uncertainty Estimation

An important question for evaluating uncertainty in the context of pre-trained models is to find out what data the model is not familiar with, and verify that it does not make overconfident wrong predictions. Given that the dataset is a black box, it is not straightforward to find a dataset that we can confidently say the model has not seen. Instead of attempting to find such a dataset, we could take inspiration from the adversarial example literature and quantify how difficult it is to produce plausible but incorrectly classified inputs.

For example, one can define an adversarial optimisation target based on the concept of feature collapse, as discussed in Chapters 3 and 4. We introduce an additive mask Δ in input space, a weight λ, and minimise the following objective for $i \neq j$:

$$\min_{\Delta} |f_\theta(\boldsymbol{x}_i + \Delta) - f_\theta(\boldsymbol{x}_j)|_2 + \lambda |\Delta|_2. \tag{6.1}$$

Intuitively, this objective aims to find the smallest change in input space such that the feature representation of $\boldsymbol{x}_i + \Delta$ and $\boldsymbol{x}_j$ have a low L2 distance. Subsequently, we can define a comparison metric based on the empirical Lipschitz constant:

$$\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}} \left[\sqrt{|\boldsymbol{x}_i| / |f_\theta(\boldsymbol{x}_i)|} \frac{|f_\theta(\boldsymbol{x}_i + \Delta) - f_\theta(\boldsymbol{x}_j)|_2}{|(\boldsymbol{x}_i + \Delta) - \boldsymbol{x}_j|_2} \right]. \tag{6.2}$$

The root is a normalisation factor to account for the difference in input and feature dimensionality for different datasets and models. As the final step of $f_\theta(\boldsymbol{x})$ and as preprocessing for $\boldsymbol{x}$, we divide by a running mean and standard deviation to make the metric scale invariant. We can use this metric to compare the extent of feature collapse robustness in a particular pre-trained model. The main caveat is that it is unclear if restricting the norm of Δ is meaningful, as this implies we are computing an L2 distance in input space, which is not a good measure of similarity.

Avoiding Feature Collapse Recent research has shown that finetuning a pretrained model on a new dataset can lead to poor OOD detection (Kumar et al., 2022). We now propose a possible way to mitigate this feature collapse by using an additional loss term. With f_θ the pretrained large model and g_ϕ the classification model placed on top. $\boldsymbol{X}$ is a minibatch of $2N$ data points with labels $\boldsymbol{y}$, ϵ and η are small step sizes:

$$\boldsymbol{X}' \leftarrow \boldsymbol{X} - \epsilon * \nabla_X |f_\theta(\boldsymbol{X}_{1:N}) - f_\theta(\boldsymbol{X}_{N:2N})|_2 \tag{6.3}$$

$$\mathcal{L} \leftarrow \mathrm{CE}(g_\phi(f_\theta(\boldsymbol{X}')), \boldsymbol{y}) - \lambda \frac{|f_\theta(\boldsymbol{X}'_{1:N}) - f_\theta(\boldsymbol{X}'_{N:2N})|_2}{|\boldsymbol{X}'_{1:N} - \boldsymbol{X}'_{N:2N}|_2} \tag{6.4}$$

$$\theta, \phi \leftarrow \theta, \phi - \eta \nabla_{\theta,\phi} \mathcal{L}, \tag{6.5}$$

with CE the cross-entropy loss function.

Intuitively, $\boldsymbol{X}'$ is one step in the direction of feature collapse between individual elements of the first half of the batch with elements of the second half. In terms of feature space distance:

$$|f_\theta(\boldsymbol{X}_{1:N}) - f_\theta(\boldsymbol{X}_{N:2N})|_2 \geq |f_\theta(\boldsymbol{X}'_{1:N}) - f_\theta(\boldsymbol{X}'_{N:2N})|_2. \tag{6.6}$$

Subsequently, we minimise the standard cross-entropy loss while maximising the feature space distance of $\boldsymbol{X}'_{1:N}$ and $\boldsymbol{X}'_{N:2N}$. An open question is what data should be used to compute and evaluate this loss term. One option is to take inspiration from the function-space VI literature and use interpolated points in pixel space (Rudner et al., 2020). Another open question is whether simply splitting a minibatch in two parts is a reasonable approach. There is significant literature on self-supervised learning that focusses on finding the best pairs of points to contrast (Robinson et al., 2020).

6.3.2 Uncertainty Estimation for In-Context Learning

By combining large vision and text models, it has become possible to classify images by simply prompting the model in the right way (Alayrac et al., 2022). For example, one can prompt the model with images of three different dogs, for each dog specifying its species "This is an image of a ..." and ending with an

uncaptioned image of a new dog. The model is then asked to predict the next word, and hopefully predicts the right class for the new dog. The examples used for the prompt can be found by searching for labelled similar images based on the feature representation of test input (Yang et al., 2021). Excitingly, the set of classes does not need to be known upfront, but the model can be queried on virtually any class. This requires a complete rethinking of the concept of "open set recognition" (as discussed in Section 2.3.3), as basically any class is now viable. This type of learning is called "in-context learning" to differentiate it from "gradient-based learning", which for example includes finetuning.

In-context learning opens up interesting new questions for uncertainty estimation. Given that there is no predictive distribution over classes, one cannot compute a form of entropy or look at the predicted probability. It is also not clear how to compute a feature space distance, as it is unclear what to use as feature representation for a class. Meanwhile, it is known that the ordering of the elements of the prompt and the content used in the prompt has a significant effect on the predictive performance (Lu et al., 2021). Even when an uncertainty estimation method is found, evaluating it will require crafting prompts where we know the model cannot do better than random guessing. In summary, this type of learning will require new mechanisms for evaluating uncertainty that go beyond what is discussed in this thesis.

6.3.3 Active Learning with Large Pre-Trained Models

Active Learning (AL) is an exciting application for large pre-trained models. Given a model that is trained on a large dataset relevant to the active learning dataset, we can expect the model to obtain excellent performance with just a few labelled examples. Due to large pre-trained models being significantly more general and robust than previous generation models (Radford et al., 2021), the chances that the model can transfer to the AL dataset is substantial. This is the scenario we considered in Section 5.6.

However, we must consider a second scenario, where the model was not pre-trained on any data relevant to the AL task. This can happen when, e.g., the AL

task is a specialised medical dataset. In this situation, it is possible that using a pre-trained model will lead to lower performance than what can be achieved with a randomly initialised model (Ash et al., 2020). Unfortunately, due to the dataset of the pre-trained model often being a black box, it is not possible to tell if a model would do well up front. Given that the AL budget can only be spent once, this complicates using pre-trained models in practice. One could use a labelled dataset similar to the AL task and evaluate AL performance there before using it on the dataset of interest, but such a dataset might not always be readily available. Not being able to predict performance up front is a big limitation in doing active learning with pre-trained models. It is currently an open question how and whether this shortcoming can be mitigated. One avenue of research could be to analyse the feature space for an unlabelled dataset, for example by using the metric suggested Section 6.3.1.

6.4 Conclusion

This thesis studies uncertainty estimation for classification, regression, and active learning in the context of deep learning. We motivated the need for uncertainty estimation in Chapter 1, and provided context on how it is quantified, used, and evaluated, in Chapter 2. We developed new mechanisms to be able to estimate uncertainty in a single forward pass in Chapter 3, and introduced a simplified extension in Section 3.4. The extension also analysed in detail the question of disentangling epistemic and aleatoric uncertainty including a benchmark to evaluate it. We subsequently placed the mechanisms of Chapter 3 in the theoretical framework of Deep Kernel Learning in Chapter 4, and showed its efficacy in active learning for causal inference in Section 4.5. In Chapter 5, we successfully argued the case that uncertainty-based batch active learning needs specific acquisition functions that focus on joint informativeness and introduced a computationally tractable way to do this that obtains excellent performance.

The work of this thesis provides a compelling framework for single forward pass uncertainty estimation. With the advent of very large models, this has now become

the default method of uncertainty estimation in deep learning. Several works have already built on this approach, such as SNGP, NUQ, and others (Antorán et al., 2020; Liu et al., 2020; Postels et al., 2020; Kotelevskii et al., 2022). A difficult open question is how to effectively and rigorously evaluate uncertainty estimation, especially in the context of models pre-trained on a private dataset. An answer to this question will enable practitioners to better evaluate their systems, and catch issues before they impact users. Since the introduction of BatchBALD in Chapter 5, several works have focused on improving it, for example BADGE (Ash et al., 2019) and PowerBALD (Kirsch et al., 2021).

As ML becomes used in more and higher stakes applications, uncertainty estimation will be a crucial component in making sure these systems are safe and dependable. We hope that improvements in large-scale modeling and uncertainty estimation go hand in hand and will lead to a bright future enabled by AI.

Appendices

A

Appendix for Chapter 3: DUQ

A.1 Experimental Details

This section contains the experimental details of Chapter 3. In all experiments, we initialise the centroids using draws from $N(0, 0.05I_n)$. All convolutional layers are initialised using the default in PyTorch 0.4.2 based on He et al. (2015).

A.1.1 Two Moons

We set the noise level to 0.1 and generate 1000 points for our training set. Our model consists of three layers with 20 hidden units each, the embedding size is 10. We use the ReLU activation function and standard SGD optimiser with learning rate 0.01, momentum 0.9 and L2 regularisation with weight 10^{-4}. Our batch size is 64, and we train for a set 30 epochs. We set the length scale to 0.3, γ to 0.99 and λ to 1.0.

A.1.2 FashionMNIST

We use a model consisting of three convolutional layers of 64, 128 and 128 3x3 filters, with a fully connected layer of 256 hidden units on top. The embedding size is 256. After every convolutional layer, we perform batch normalisation and a 2x2 max pooling operation.

We use the SGD optimiser with learning rate 0.05 (decayed by a factor of 5 every 10 epochs), momentum 0.9, weight decay 10^{-4} and train for a set 30 epochs. The centroid updates are done with $\gamma = 0.999$. The output feature dimension of the model is 256, and we use the same value for the dimensionality of the centroids.

We normalise our data using per channel mean and standard deviation, as computed on the training set. The validation set contains 5000 elements, removed at random from the full 60,000 elements in the training set. For the final results, we re-train on the full training set with the final set of hyperparameters.

A.1.3 CIFAR-10

We use a ResNet-18, as implement in TorchVision version 0.4.2. With the following modifications: the first convolutional layer is changed to have 64 3x3 filters with stride 1, the first pooling layer is skipped, and the last linear layer is changed to be 512 by 512.

We use the SGD optimiser with learning rate of 0.05, decayed by a factor 10 every 25 epochs, momentum of 0.9, weight decay 10^{-4}, and we train for a set 75 epochs. The centroid updates are done with $\gamma = 0.999$. The output dimension of the model, d is 512, and we use the same value for the size of the centroids n.

We normalise our data using per channel mean and standard deviation, as computed on the training set. We augment the data at training time using random horizontal flips (with probability 0.5) and random crops after padding 4 zero pixels on all sides. The validation set contains 10,000 elements, removed at random from the full 50,000 elements in the training set. For the final results, we re-train on the full training set with the final set of hyperparameters.

B

Appendix for Chapter 4: DUE

B.1 Experimental Details

This section lists the experimental details for all experiments in Chapter 4.

B.1.1 1D and 2D Experiments

We perform these experiments using a simple feed-forward ResNet, similar to Liu et al. (2020). The first linear layer maps from the input to the initial feature representation and does not have an activation function. After which the model is a fully connected ResNet consisting of blocks computing $x' = x + f(x)$, with $f(\cdot)$ a combination of a linear mapping and a ReLU activation function. Spectral normalisation is applied to the linear mapping for which we use the implementation of Behrmann et al. (2019). We use the Adam optimiser for regression and SGD for the two moons classification with learning rate 0.01. We use 4 layers with 128 features, a maximum spectral norm of 0.95, a single power iteration to compute the spectral norm at each step, and an RBF kernel. Additive Gaussian noise with standard deviation 0.1 is added to samples from the two moons dataset. For the toy regression experiment, we use 20 inducing points and for two moons we use 4.

B.1.2 CIFAR-10

For the WRN, we follow the experimental setup and implementation of Zagoruyko et al. (2016). This means that for CIFAR-10, we use depth 28 with widen factor 10 and BasicBlocks with dropout. We train for the prescribed 200 epochs (no early stopping) with batch size 128, starting with learning rate 0.1 and dropping with a factor of 0.2 at epoch 60, 120 and 160. SGD is set to use a momentum of 0.9 and weight decay 5e-4. We select 20% of the training data at random as a validation set for hyperparameter tuning, but obtain the final model by using the full training set with the final set of hyperparameters. SV-DKL was trained using the suggested values in the GPyTorch (Gardner et al., 2018) tutorial: grid size set to 64 and grid bounds between -10 and 10.

For spectral normalisation, we use the implementation of Behrmann et al. (2019), and also constrain batch normalisation as described in Gouk et al. (2018). We use 1 power iteration and use the lowest spectral norm that still allows for good accuracy, which we found to be around 3 in practice. We set the momentum of spectral batch normalisation to 0.99 to reduce the variance of the running average estimator of the empirical feature variance, which can be a source of instability.

B.1.3 Regression Experiment

Following Shalit et al. (2017), we use 63%/27%/10% train / validation / test splits and report the RMSE (or equivalently PEHE) evaluated on the test set over 1000 trials. For each trial, we train for a maximum of 750 epochs with batch size 100 and report results on the model with the lowest negative log-likelihood evaluated on the validation set. We employ Adam optimisation with a learning rate of 0.001 and batch size of 100.

The feature extractor uses a feed-forward ResNet architecture with 3 layers, 200 hidden units per layer, and ELU activations. Dropout is applied after each activation at a rate of 0.1. The feature extractor takes the individual x as input, and treatment t is appended to the output of the feature extractor, in similar fashion to the TARNet architecture. Spectral normalisation with value 0.95 is

used on all layers of the feature extractor. The output GP uses a Matérn kernel with $\nu = \frac{3}{2}$ and 100 inducing points.

The baselines in Table 4.5 are: BART which is based on decision trees (Chipman et al., 2010), BCEVAE which is the MC Dropout extension of CEVAE (Louizos et al., 2017), and DKLITE (Zhang et al., 2020) which is a recent method also based on DKL. Results are obtained from Jesson et al. (2020). DKLITE works by jointly training a VAE (Kingma et al., 2013) with a GP on the latent dimensions of the VAE. For the DKLITE experiments we use the open source code from the authors, available from bitbucket. We write a custom loop over the IHDP dataset to follow the above protocol. BTARNet, ensemble of TARNet and DUE all use the same feature extractor, as detailed in Appendix B.1. We report the root mean squared error between the predicted CATE and the true CATE (which is given in the dataset) to assess the error on the held out dataset (lower is better).

B.1.4 Gaussian Process Initialisation

The inducing points locations are initialised using centroids obtained from performing k-means on the feature representation of 1,000 points randomly chosen from the training set. The initial length scales are computed by taking the average pairwise euclidean distance between feature representation, as computed using a randomly initialised model, of the 1,000 points. The GP is implemented using GPyTorch (Gardner et al., 2018) and uses their default values if not otherwise specified.

B.1.5 Spectral Normalisation

We now describe the spectral normalisation procedure used in Section 3.4 and Chapter 4. It is based on Miyato et al. (2018) and Gouk et al. (2018). The implementation for the experiments in this paper is based on the implementation of Behrmann et al. (2019).

The goal of spectral normalisation is to constrain the maximum singular value. The maximum singular value can be determined exactly by SVD, but this is too computationally expensive to run at every forward step. Instead we depend on

Algorithm 4 Spectral Normalization for a linear layer

- Initialize $\boldsymbol{u}_l \in \mathcal{R}^{d_l}$. With d the number of hidden units in layer l for $l = 1, \ldots, L$ with a random vector (sampled from an isotropic Gaussian distribution).
- For each weight matrix $\boldsymbol{W}_l$ at layer l:

1. Apply power iteration method to a unnormalized weight W_l:

$$\boldsymbol{v}_l \leftarrow \boldsymbol{W}_l^{\mathrm{T}} \boldsymbol{u}_l / \|(\boldsymbol{W}_l)^{\mathrm{T}} \boldsymbol{u}_l\|_2 \tag{B.1}$$

$$\boldsymbol{u}_l \leftarrow \boldsymbol{W}_l \boldsymbol{v}_l / \|\boldsymbol{W}_l \boldsymbol{v}_l\|_2 \tag{B.2}$$

2. Calculate W_{SN} with the spectral norm:

$$\boldsymbol{W}_l^{\mathrm{SN}}(\boldsymbol{W}_l) = \boldsymbol{W}_l / \sigma(\boldsymbol{W}_l), \text{ where } \sigma(\boldsymbol{W}_l) = \boldsymbol{u}_l^{\mathrm{T}} \boldsymbol{W}_l \boldsymbol{v}_l \tag{B.3}$$

the fact that the value only changes a small amount at every gradient update and estimate it with online power iteration, see Algorithm 4.

Based on Gouk et al. (2018), Algorithm 4 can be extended to convolutions by simply relying on the convolution and transposed convolution operator instead of the matrix multiplications between $\boldsymbol{W}$ and $\boldsymbol{u}_l, \boldsymbol{v}_l$.

We additionally constrain batch normalisation as discussed in Section 4.2.2 and we repeat it here for completeness. Batch normalisation transforms the input following:

$$x_{out} = \mathrm{diag}\left(\frac{\gamma}{\sqrt{\mathrm{Var}(x)}}\right)(x - \mathbb{E}[x]) + \beta, \tag{B.4}$$

with γ and β the learnable scale and shift parameters. It has a Lipschitz constant of $\max_i |\frac{\gamma_i}{\sqrt{\mathrm{Var}(x)}_i}|$ (Gouk et al., 2018). Using the above equation, we can extend spectral normalisation to batch normalisation by dividing the weight γ of the batch normalisation by the (scaled) Lipschitz constant:

$$\gamma_{\mathrm{SN}} = \frac{\gamma}{\max_i |\frac{\gamma_i}{\sqrt{\mathrm{Var}(x)}_i}|} \tag{B.5}$$

These procedures limit the maximum singular value to 1. In practice, we find that this value can be too constraining and instead use a larger coefficient (similar

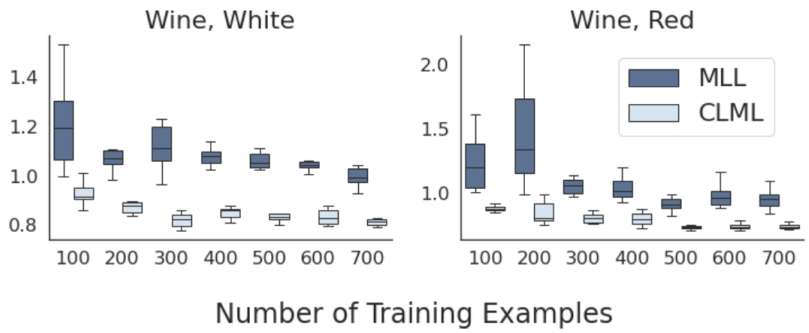


Figure B.1: Results of applying DUE with LML and CLML to two UCI datasets. We see that CLML improves performance compared to LML, and that DUE's insights further improve upon the results in Figure 7a. of Lotfi et al. (2022).

to Liu et al. (2020)). We then normalize the weights such that their maximum singular value is at most the larger coefficient.

B.2 Conditional Log Marginal Likelihood

We evaluate if the findings to avoid feature collapse in DKL discussed in Chapter 4 are complimentary with the method of using the conditional log marginal likelihood (CLML) (Lotfi et al., 2022). For this experiment, we use the same feature extractor as in Section B.1.1, with spectral normalisation and coefficient set to 0.95 as in the 1D and 2D experiment in Chapter 4. We apply it following the experimental setup of Lotfi et al. (2022) to the Wine datasets of UCI. In comparison with Figure 7 (a) of Lotfi et al. (2022), we see that DUE works complimentary with CLML and further improves the performance.

C

Appendix for Chapter 5: BatchBALD

C.1 Experimental Details

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

EMNIST We use a similar model architecture as on MNIST, 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.

C.2 Proof of Submodularity

Nemhauser et al. (1978) show that if a function is submodular, then a greedy algorithm like Algorithm 3 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 C.2.1. *A function f defined on subsets of Ω is called* submodular *if for every set $A \subset \Omega$ and two non-identical points $y_1, y_2 \in \Omega \setminus A$:*

$$f(A \cup \{y_1\}) + f(A \cup \{y_2\}) \geq f(A \cup \{y_1, y_2\}) + f(A) \tag{C.1}$$

Intuitively, submodularity expresses that there are "diminishing returns" for adding additional points to A.

Lemma C.2.2. $a_{\texttt{BatchBALD}}(A, \mathrm{p}(\theta)) := \mathbb{I}(A\,;\,\theta)$ *is submodular for $A \subset \mathcal{D}_{\text{pool}}$.*

Proof. Let $y_1, y_2 \in \mathcal{D}_{\text{pool}}, y_1 \neq y_2$. We start by substituting the definition of $a_{\texttt{BatchBALD}}$ into Equation C.1 and subtracting $\mathbb{I}(A\,;\,\theta)$ twice on both sides, using the fact that $\mathbb{I}(A \cup B\,;\,\theta) - \mathbb{I}(B\,;\,\theta) = \mathbb{I}(A\,;\,\theta \mid B)$:

$$\mathbb{I}(A \cup \{y\}\,;\,\theta) + \mathbb{I}(A \cup \{x\}\,;\,\theta) \geq \mathbb{I}(A \cup \{x, y\}\,;\,\theta) + \mathbb{I}(A\,;\,\theta) \tag{C.2}$$

$$\Leftrightarrow\ \mathbb{I}(y\,;\,\theta \mid A) + \mathbb{I}(x\,;\,\theta \mid A) \geq \mathbb{I}(x, y\,;\,\theta \mid A). \tag{C.3}$$

We rewrite the left-hand side using the definition of the mutual information $\mathbb{I}(A\,;\,B) = \mathbb{H}(A) - \mathbb{H}(A \mid B)$ and reorder:

$$\mathbb{I}(y\,;\,\theta \mid A) + \mathbb{I}(x\,;\,\theta \mid A) \tag{C.4}$$

$$= \underbrace{\mathbb{H}(y_1 \mid A) + \mathbb{H}(y_2 \mid A)}_{\geq \mathbb{H}(y_1, y_2 \mid A)} - \underbrace{(\mathbb{H}(y_1 \mid A, \theta) + \mathbb{H}(y_2 \mid A, \theta))}_{= \mathbb{H}(y_1, y_2 \mid A, \theta)} \tag{C.5}$$

$$\geq \mathbb{H}(y_1, y_2 \mid A) - \mathbb{H}(y_1, y_2 \mid A, \theta) \tag{C.6}$$

$$= \mathbb{I}(x, y\,;\,\theta \mid A), \tag{C.7}$$

where we have used that entropies are subadditive in general and additive given $y_1 \perp\!\!\!\perp y_2 \mid \theta$. □

Following Nemhauser et al. (1978), we can conclude that Algorithm 3 is $1 - 1/e$-approximate.

C.3 BatchBALD as an Approximation of BALD with Acquisition Size 1

To see why BALD with acquisition size 1 can be seen as an upper bound for BatchBALD performance in an idealised setting, we reformulate line 5 in Algorithm 3 on page 88. Instead of the original term $a_{\texttt{BatchBALD}}\left(A_{n-1} \cup \{\boldsymbol{x}\}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})\right)$, we can equivalently maximise:

$$a_{\texttt{BatchBALD}}\left(A_{n-1} \cup \{\boldsymbol{x}\}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})\right) - a_{\texttt{BatchBALD}}\left(A_{n-1}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}})\right) \tag{C.8}$$

as the right term is constant for all $\boldsymbol{x} \in \mathcal{D}_{\text{pool}} \setminus A_{n-1}$. In turn, this is equivalent to:

$$= \mathbb{I}(y_1, ..., y_{n-1}, y\,;\, \theta \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}, \boldsymbol{x}, \mathcal{D}_{\text{train}}) - \mathbb{I}(y_1, ..., y_{n-1}\,;\, \theta \mid \boldsymbol{x}_{1:n-1}\mathcal{D}_{\text{train}}) \tag{C.9}$$

$$= \mathbb{I}(y\,;\, \theta \mid \boldsymbol{x}, y_1, ..., y_{n-1}, \boldsymbol{x}_{1:n-1}, \mathcal{D}_{\text{train}}) \tag{C.10}$$

once we expand $A_{n-1} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}\}$. At each step of the inner loop, our greedy algorithm is maximising the mutual information of the individual available data points with the model parameters conditioned on all the additional data points that have already been picked for acquisition and the existing training set. Finally, assuming training our model captures all available information,

$$\geq \mathbb{I}(y\,;\, \theta \mid \boldsymbol{x}, \mathcal{D}_{\text{train}} \cup \{(\boldsymbol{x}_1, \mathrm{y}_1), ..., (\boldsymbol{x}_{n-1}, \tilde{y}_{n-1})\}) \tag{C.11}$$

$$= a_{\texttt{BALD}}\left(\{\boldsymbol{x}\}, \mathrm{p}(\theta \mid \mathcal{D}_{\text{train}} \cup \{(\boldsymbol{x}_1, \mathrm{y}_1), ..., (\boldsymbol{x}_{n-1}, \tilde{y}_{n-1})\})\right), \tag{C.12}$$

where $\tilde{y}_1, ..., \tilde{y}_{n-1}$ are the actual labels of $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$. The mutual information decreases as θ becomes more concentrated as we expand the training set size, and thus the overlap of y and θ will become smaller (in an information-measure-theoretical sense).

This shows that every step n of the inner loop in our algorithm is at most as good as retraining our model on the new training set $\mathcal{D}_{\text{train}} \cup \{(\boldsymbol{x}_1, \mathrm{y}_1), ..., (\boldsymbol{x}_{n-1}, \tilde{y}_{n-1})\}$ and picking x_n using $a_{\texttt{BALD}}$ with acquisition size 1.

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.

C.4 Sampling of Configurations

We are using the same notation as in Section 5.2.2. We factor $\mathrm{p}(y_{1:n} \mid \theta)$ to avoid duplicating computation and rewrite $\mathbb{H}(y_{1:n})$ as:

$$\mathbb{H}(y_{1:n}) = \mathbb{E}_{\mathrm{p}(\theta)}\, \mathbb{E}_{\mathrm{p}(y_{1:n}|\theta)} \left[-\log \mathrm{p}(y_{1:n}) \right] \tag{C.13}$$

$$= \mathbb{E}_{\mathrm{p}(\theta)}\, \mathbb{E}_{\mathrm{p}(y_{1:n-1}|\theta)\, \mathrm{p}(y_n|\theta)} \left[-\log \mathrm{p}(y_{1:n}) \right] \tag{C.14}$$

$$= \mathbb{E}_{\mathrm{p}(\theta)}\, \mathbb{E}_{\mathrm{p}(y_{1:n-1}|\theta)}\, \mathbb{E}_{\mathrm{p}(y_n|\theta)} \left[-\log \mathrm{p}(y_{1:n}) \right] \tag{C.15}$$

To be flexible in the way we sample $y_{1:n-1}$, we perform importance sampling of $\mathrm{p}(y_{1:n-1} \mid \theta)$ 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:

$$\mathbb{H}(y_{1:n}) = \mathbb{E}_{\mathrm{p}(\theta)}\, \mathbb{E}_{\mathrm{p}(y_{1:n-1})} \left[\frac{\mathrm{p}(y_{1:n-1} \mid \theta)}{\mathrm{p}(y_{1:n-1})}\, \mathbb{E}_{\mathrm{p}(y_n|\theta)} \left[-\log \mathrm{p}(y_{1:n}) \right] \right] \tag{C.16}$$

$$= \mathbb{E}_{\mathrm{p}(y_{1:n-1})}\, \mathbb{E}_{\mathrm{p}(\theta)}\, \mathbb{E}_{\mathrm{p}(y_n|\theta)} \left[-\frac{\mathrm{p}(y_{1:n-1} \mid \theta)}{\mathrm{p}(y_{1:n-1})} \log \mathbb{E}_{\mathrm{p}(\theta)} \left[\mathrm{p}(y_{1:n-1} \mid \theta)\, \mathrm{p}(y_{1:n} \mid \theta) \right] \right] \tag{C.17}$$

$$\approx -\frac{1}{m} \sum_{\hat{y}_{1:n-1}}^{m} \sum_{\hat{y}_n} \frac{\frac{1}{k} \sum_{\hat{\boldsymbol{w}}_j} \mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{w}}_j)\, \mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{w}}_j)}{\mathrm{p}(\hat{y}_{1:n-1})} \log \left[\frac{1}{k} \sum_{\hat{\boldsymbol{w}}_j} \mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{w}}_j)\, \mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{w}}_j) \right] \tag{C.18}$$

$$= -\frac{1}{m} \sum_{\hat{y}_{1:n-1}}^{m} \sum_{\hat{y}_n} \frac{\left(\hat{P}_{1:n-1} \hat{P}_n^T \right)_{\hat{y}_{1:n-1}, \hat{y}_n}}{\left(\hat{P}_{1:n-1} \mathbb{1}_{k,1} \right)_{\hat{y}_{1:n-1}}} \log \left(\frac{1}{k} \left(\hat{P}_{1:n-1} \hat{P}_n^T \right)_{\hat{y}_{1:n-1}, \hat{y}_n} \right), \tag{C.19}$$

where we store $\mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{w}}_j)$ in a matrix $\hat{P}_{1:n-1}$ of shape $m \times k$ and $\mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{w}}_j)$ in a matrix $\hat{P}_n$ of shape $c \times k$ and $\mathbb{1}_{k,1}$ is a $k \times 1$ matrix of 1s. Equation C.19 allows us to cache $\hat{P}_{1:n-1}$ inside the inner loop of Algorithm 3 and use batch matrix multiplication for efficient computation.

References

Abrevaya, Jason, Yu-Chin Hsu, and Robert P Lieli (2015). "Estimating conditional average treatment effects". In: *Journal of Business & Economic Statistics* 33.4, pp. 485–505.

Adomavicius, Gediminas and Alexander Tuzhilin (2005). "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions". In: *IEEE Transactions on Knowledge & Data Engineering*.

Aitchison, Laurence (2020). "A statistical theory of cold posteriors in deep neural networks". In: *International Conference on Learning Representations*.

Alayrac, Jean-Baptiste, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. (2022). "Flamingo: a visual language model for few-shot learning". In: *arXiv preprint arXiv:2204.14198*.

Alvi, Ahsan S, Binxin Ru, Jan Calliess, Stephen J Roberts, and Michael A Osborne (2019). "Asynchronous Batch Bayesian Optimisation with Improved Local Penalisation". In: *arXiv preprint arXiv:1901.10452*.

Antorán, Javier, James Allingham, and José Miguel Hernández-Lobato (2020). "Depth uncertainty in neural networks". In: *Advances in neural information processing systems* 33, pp. 10620–10634.

Ash, Jordan and Ryan P Adams (2020). "On warm-starting neural network training". In: *Advances in Neural Information Processing Systems* 33, pp. 3884–3894.

Ash, Jordan T, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal (2019). "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds". In: *International Conference on Learning Representations*.

Azimi, Javad, Alan Fern, Xiaoli Zhang-Fern, Glencora Borradaile, and Brent Heeringa (2012). "Batch active learning via coordinated matching". In: *arXiv preprint arXiv:1206.6458*.

Band, Neil, Tim G. J. Rudner, Qixuan Feng, Angelos Filos, Zachary Nado, Michael W Dusenberry, Ghassen Jerfel, Dustin Tran, and Yarin Gal (2021). "Benchmarking Bayesian Deep Learning on Diabetic Retinopathy Detection Tasks". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Behrmann, Jens, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen (2019). "Invertible residual networks". In: *International Conference on Machine Learning*, pp. 573–582.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight Uncertainty in Neural Network". In: *International Conference on Machine Learning*, pp. 1613–1622.

Bradshaw, John, Alexander G de G Matthews, and Zoubin Ghahramani (2017). "Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks". In: *arXiv preprint arXiv:1707.02476*.

Breiman, Leo (2001). "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3, pp. 199–231.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). "Language models are few-shot learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Bulatov, Yaroslav (2011). "notMNIST dataset". In: *Tech. Rep.[Online]. Available: http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html*.

Burt, David, Carl Edward Rasmussen, and Mark van der Wilk (2019). "Rates of convergence for sparse variational Gaussian process regression". In: *International Conference on Machine Learning*. PMLR, pp. 862–871.

Calandra, Roberto, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth (2016). "Manifold Gaussian processes for regression". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 3338–3345.

Calinon, Sylvain, Florent Guenter, and Aude Billard (2007). "On learning, representing, and generalizing a task in a humanoid robot". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37.2, pp. 286–298.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2020). "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR, pp. 1597–1607.

Chipman, Hugh A, Edward I George, Robert E McCulloch, et al. (2010). "BART: Bayesian additive regression trees". In: *The Annals of Applied Statistics* 4.1, pp. 266–298.

Cohen, Gregory, Saeed Afshar, Jonathan Tapson, and André van Schaik (2017). "EMNIST: Extending MNIST to handwritten letters". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 2921–2926.

Cohn, David A, Zoubin Ghahramani, and Michael I Jordan (1996). "Active learning with statistical models". In: *Journal of artificial intelligence research* 4, pp. 129–145.

Cuong, Nguyen Viet, Wee Sun Lee, Nan Ye, Kian Ming A Chai, and Hai Leong Chieu (2013). "Active learning for probabilistic hypotheses using the maximum Gibbs error criterion". In: *Advances in Neural Information Processing Systems*, pp. 1457–1465.

D'Amour, Alexander, Peng Ding, Avi Feller, Lihua Lei, and Jasjeet Sekhon (2021). "Overlap in observational studies with high-dimensional covariates". In: *Journal of Econometrics* 221.2, pp. 644–654.

D'Angelo, Francesco and Vincent Fortuin (2021). "Repulsive deep ensembles are bayesian". In: *Advances in Neural Information Processing Systems* 34.

Darlow, Luke N, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey (2018). "CINIC-10 is not ImageNet or CIFAR-10". In: *arXiv preprint arXiv:1810.03505*.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.

Dinh, Laurent, David Krueger, and Yoshua Bengio (2014). "Nice: Non-linear independent components estimation". In: *arXiv preprint arXiv:1410.8516*.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2016). "Density estimation using real nvp". In: *arXiv preprint arXiv:1605.08803*.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). "An Image

is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*.

Drucker, Harris and Yann Le Cun (1992). "Improving generalization performance using double backpropagation". In: *IEEE Transactions on Neural Networks* 3.6, pp. 991–997.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*.

Filos, Angelos, Sebastian Farquhar, Aidan N Gomez, Tim GJ Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal (2019). "A Systematic Comparison of Bayesian Deep Learning Robustness in Diabetic Retinopathy Tasks". In: *arXiv preprint arXiv:1912.10481*.

Finlayson, Samuel G, Adarsh Subbaswamy, Karandeep Singh, John Bowers, Annabel Kupke, Jonathan Zittrain, Isaac S Kohane, and Suchi Saria (2021). "The clinician and dataset shift in artificial intelligence". In: *The New England journal of medicine* 385.3, p. 283.

Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan (2019). "Deep ensembles: A loss landscape perspective". In: *arXiv preprint arXiv:1912.02757*.

Frankle, Jonathan and Michael Carbin (2019). "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*.

Freeman, Linton C (1965). *Elementary applied statistics: for students in behavioral science*. John Wiley & Sons.

Gal, Yarin (2016). "Uncertainty in deep learning". In: *University of Cambridge* 1, p. 3.

Gal, Yarin and Zoubin Ghahramani (2016). "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". In: *International Conference on Machine Learning*, pp. 1050–1059.

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani (2017). "Deep bayesian active learning with image data". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1183–1192.

Galil, Ido, Mohammed Dabbah, and Ran El-Yaniv (2022). *How to measure deep uncertainty estimation performance and which models are naturally better at providing it*.

Gardner, Jacob R, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson (2018). "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration". In: *Advances in Neural Information Processing Systems*.

Ghahramani, Zoubin (2015). "Probabilistic machine learning and artificial intelligence". In: *Nature* 521.7553, pp. 452–459.

González, Javier, Zhenwen Dai, Philipp Hennig, and Neil Lawrence (2016). "Batch Bayesian optimization via local penalization". In: *Artificial Intelligence and Statistics*, pp. 648–657.

Gouk, Henry, Eibe Frank, Bernhard Pfahringer, and Michael Cree (2018). "Regularisation of neural networks by enforcing lipschitz continuity". In: *arXiv preprint arXiv:1804.04368*.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville (2017). "Improved training of wasserstein gans". In: *Advances in neural information processing systems*, pp. 5767–5777.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger (2017). "On calibration of modern neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 1321–1330.

Guo, Yuhong and Dale Schuurmans (2008). "Discriminative batch mode active learning". In: *Advances in neural information processing systems*, pp. 593–600.

He, Bobby, Balaji Lakshminarayanan, and Yee Whye Teh (2020). "Bayesian deep ensembles via the neural tangent kernel". In: *Advances in Neural Information Processing Systems* 33, pp. 1010–1022.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

— (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hein, Matthias, Maksym Andriushchenko, and Julian Bitterwolf (2019). "Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 41–50.

Hendrycks, Dan, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. (2021). "The many faces of robustness: A critical analysis of out-of-distribution generalization". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349.

Hendrycks, Dan and Kevin Gimpel (2017). "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *International Conference on Learning Representations*.

Hensman, James, Nicolas Durrande, and Arno Solin (2017). "Variational Fourier features for Gaussian processes". In: *The Journal of Machine Learning Research* 18.1, pp. 5537–5588.

Hensman, James, Alexander Matthews, and Zoubin Ghahramani (2015). "Scalable variational Gaussian process classification". In: *JMLR*.

Hernández-Lobato, Daniel, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams (2016). "Predictive entropy search for multi-objective Bayesian optimization". In: *International Conference on Machine Learning*, pp. 1492–1501.

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017). "beta-VAE: Learning basic visual concepts with a constrained variational framework". In.

Hill, Jennifer L (2011). "Bayesian nonparametric modeling for causal inference". In: *Journal of Computational and Graphical Statistics* 20.1, pp. 217–240.

Hinton, Geoffrey E, Simon Osindero, and Yee-Whye Teh (2006). "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7, pp. 1527–1554.

Hinton, Geoffrey E and Russ R Salakhutdinov (2008). "Using deep belief nets to learn covariance kernels for Gaussian processes". In: *Advances in neural information processing systems*, pp. 1249–1256.

Hoffman, Judy, Daniel A Roberts, and Sho Yaida (2019). "Robust learning with jacobian regularization". In: *arXiv preprint arXiv:1908.02729*.

Hoi, Steven CH, Rong Jin, Jianke Zhu, and Michael R Lyu (2006). "Batch mode active learning and its application to medical image classification". In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 417–424.

Houlsby, Neil, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel (2011). "Bayesian active learning for classification and preference learning". In: *arXiv preprint arXiv:1112.5745*.

Hutchinson, Michael F (1990). "A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines". In: *Communications in Statistics-Simulation and Computation* 19.2, pp. 433–450.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *International Conference on Machine Learning*, pp. 448–456.

Izmailov, Pavel, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson (2021). "What are Bayesian neural network posteriors really like?" In: *International conference on machine learning*. PMLR, pp. 4629–4640.

Jacobsen, Jörn-Henrik, Arnold WM Smeulders, and Edouard Oyallon (2018). "i-RevNet: Deep Invertible Networks". In: *International Conference on Learning Representations*.

Janz, David, Jos van der Westhuizen, and José Miguel Hernández-Lobato (2017). "Actively learning what makes a discrete sequence valid". In: *arXiv preprint arXiv:1708.04465*.

Jesson, Andrew, Sören Mindermann, Uri Shalit, and Yarin Gal (2020). "Identifying Causal-Effect Inference Failure with Uncertainty-Aware Models". In: *Advances in Neural Information Processing Systems* 33.

Jesson*, Andrew, Panagiotis Tigas*, Joost van Amersfoort, Andreas Kirsch, Uri Shalit, and Yarin Gal (2021). "Causal-BALD: Deep Bayesian Active Learning of Outcomes to Infer Treatment-Effects from Observational Data". In: *Advances in Neural Information Processing Systems* 34.

Jolicoeur-Martineau, Alexia and Ioannis Mitliagkas (2019). "Connections between Support Vector Machines, Wasserstein distance and gradient-penalty GANs". In: *arXiv preprint arXiv:1910.06922*.

Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei (2020). "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361*.

Kapoor, Sanyam, Wesley J Maddox, Pavel Izmailov, and Andrew G Wilson (2022). "On uncertainty, tempering, and data augmentation in bayesian classification". In: *Advances in Neural Information Processing Systems* 35, pp. 18211–18225.

Kendall, Alex, Vijay Badrinarayanan, and Roberto Cipolla (2015). "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding". In: *arXiv preprint arXiv:1511.02680*.

Kendall, Alex and Yarin Gal (2017). "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems* 30.

Kingma, Diederik P and Jimmy Ba (2014a). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling (2014b). "Semi-supervised learning with deep generative models". In: *Advances in neural information processing systems*, pp. 3581–3589.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kirichenko, Polina, Pavel Izmailov, and Andrew G Wilson (2020). "Why normalizing flows fail to detect out-of-distribution data". In: *Advances in neural information processing systems* 33, pp. 20578–20589.

Kirsch, Andreas, Sebastian Farquhar, Parmida Atighehchian, Andrew Jesson, Frederic Branchaud-Charron, and Yarin Gal (2021). "Stochastic Batch Acquisition for Deep Active Learning". In: *arXiv preprint arXiv:2106.12059*.

Kirsch*, Andreas, Joost van Amersfoort*, and Yarin Gal (2019). "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning". In: *Advances in Neural Information Processing Systems*, pp. 7026–7037.

Kossen, Jannik, Sebastian Farquhar, Yarin Gal, and Tom Rainforth (2021). "Active testing: Sample-efficient model evaluation". In: *International Conference on Machine Learning*. PMLR, pp. 5753–5763.

Kotelevskii, Nikita, Aleksandr Artemenkov, Kirill Fedyanin, Fedor Noskov, Alexander Fishkov, Aleksandr Petiushko, and Maxim Panov (2022). "NUQ: Nonparametric Uncertainty Quantification for Deterministic Neural Networks". In: *arXiv preprint arXiv:2202.03101*.

Krause, Andreas, Ajit Singh, and Carlos Guestrin (2008). "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies". In: *Journal of Machine Learning Research* 9.Feb, pp. 235–284.

Krizhevsky, Alex, Geoffrey Hinton, et al. (2009). "Learning multiple layers of features from tiny images". In: *Tech Report*.

Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton (2014). "The cifar-10 dataset". In: *online: http://www. cs. toronto. edu/kriz/cifar. html* 55.

Kullback, Solomon and Richard A Leibler (1951). "On information and sufficiency". In: *The annals of mathematical statistics* 22.1, pp. 79–86.

Kumar, Ananya, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang (2022). "Fine-tuning can distort pretrained features and underperform out-of-distribution". In: *arXiv preprint arXiv:2202.10054*.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in Neural Information Processing Systems*, pp. 6402–6413.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998a). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

LeCun, Yann, Corinna Cortes, and Christopher JC Burges (1998b). "The MNIST database of handwritten digits, 1998". In: *URL http://yann.lecun.com/exdb/mnist* 10, p. 34.

Lee, Kimin, Kibok Lee, Honglak Lee, and Jinwoo Shin (2018). "A simple unified framework for detecting out-of-distribution samples and adversarial attacks". In: *Advances in Neural Information Processing Systems*, pp. 7167–7177.

Liu, Jeremiah Zhe, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan (2020). "Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness". In: *Advances in Neural Information Processing Systems 33*.

Lotfi, Sanae, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson (2022). "Bayesian model selection, the marginal likelihood, and generalization". In: *International Conference on Machine Learning*. PMLR, pp. 14223–14247.

Louizos, Christos, Uri Shalit, Joris Mooij, David Sontag, Richard Zemel, and Max Welling (2017). "Causal effect inference with deep latent-variable models". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6449–6459.

Lu, Yao, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp (2021). "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity". In: *arXiv preprint arXiv:2104.08786*.

MacKay, David JC (1992). "Bayesian methods for adaptive models". PhD thesis. California Institute of Technology.

Maddox, Wesley J, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson (2019). "A simple baseline for bayesian uncertainty in deep learning". In: *Advances in neural information processing systems* 32.

Matthews, Alexander Graeme de Garis (2017). "Scalable Gaussian process inference using variational methods". PhD thesis. University of Cambridge.

Milios, Dimitrios, Raffaello Camoriano, Pietro Michiardi, Lorenzo Rosasco, and Maurizio Filippone (2018). "Dirichlet-based gaussian processes for large-scale calibrated classification". In: *Advances in Neural Information Processing Systems* 31.

Minderer, Matthias, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic (2021). "Revisiting the calibration of modern neural networks". In: *Advances in Neural Information Processing Systems* 34.

Miyato, Takeru, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida (2018). "Spectral Normalization for Generative Adversarial Networks". In: *International Conference on Learning Representations*.

Moreno-Torres, Jose G, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera (2012). "A unifying view on dataset shift in classification". In: *Pattern recognition* 45.1, pp. 521–530.

Mukhoti*, Jishnu, Andreas Kirsch*, Joost van Amersfoort, Philip HS Torr, and Yarin Gal (2023). "Deep Deterministic Uncertainty: A New Simple Baseline". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *International Conference on Machine Learning*.

Nalisnick, Eric, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan (2019a). "Do Deep Generative Models Know What They Don't Know?" In: *International Conference on Learning Representations*.

— (2019b). "Hybrid models with deep and invertible features". In: *arXiv preprint arXiv:1902.02767*.

Nalisnick, Eric, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan (2019c). "Detecting out-of-distribution inputs to deep generative models using typicality". In: *arXiv preprint arXiv:1906.02994*.

Neal, Radford M (1996). *Bayesian learning for neural networks*. Springer Science & Business Media.

Nemhauser, George L, Laurence A Wolsey, and Marshall L Fisher (1978). "An analysis of approximations for maximizing submodular set functions—I". In: *Mathematical programming* 14.1, pp. 265–294.

Netzer, Yuval, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng (2011). "Reading digits in natural images with unsupervised feature learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Noci, Lorenzo, Kevin Roth, Gregor Bachmann, Sebastian Nowozin, and Thomas Hofmann (2021). "Disentangling the Roles of Curation, Data-Augmentation and the Prior in the Cold Posterior Effect". In: *Advances in Neural Information Processing Systems* 34.

Ovadia, Yaniv, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek (2019). "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift". In: *Advances in Neural Information Processing Systems*, pp. 13991–14002.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. (2011). "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct, pp. 2825–2830.

Petzka, Henning, Asja Fischer, and Denis Lukovnicov (2017). "On the regularization of Wasserstein GANs". In: *arXiv preprint arXiv:1709.08894*.

Postels, Janis, Hermann Blum, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari (2020). "Quantifying aleatoric and epistemic uncertainty using density estimation in latent space". In: *arXiv preprint arXiv:2012.03082*.

Quinn, John A and Masashi Sugiyama (2014). "A least-squares approach to anomaly detection in static and sequential data". In: *Pattern Recognition Letters* 40, pp. 36–40.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. (2021). "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR, pp. 8748–8763.

Rahimi, Ali and Benjamin Recht (2008). "Random features for large-scale kernel machines". In: *Advances in neural information processing systems*, pp. 1177–1184.

Rasmus, Antti, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko (2015). "Semi-supervised learning with ladder networks". In: *Advances in neural information processing systems*, pp. 3546–3554.

Rasmussen, C.E. and C.K.I. Williams (2006). *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited.

Recht, Benjamin, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar (2018). "Do CIFAR-10 classifiers generalize to CIFAR-10?" In: *arXiv preprint arXiv:1806.00451*.

— (2019). "Do imagenet classifiers generalize to imagenet?" In: *International Conference on Machine Learning*. PMLR, pp. 5389–5400.

Ren, Jie, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan (2021). "A simple fix to mahalanobis distance for improving near-ood detection". In: *arXiv preprint arXiv:2106.09022*.

Ren, Jie, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan (2019). "Likelihood Ratios for Out-of-Distribution Detection". In: *Advances in Neural Information Processing Systems 32*, pp. 14680–14691.

Robinson, Joshua, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka (2020). "Contrastive learning with hard negative samples". In: *arXiv preprint arXiv:2010.04592*.

Rosca, Mihaela, Theophane Weber, Arthur Gretton, and Shakir Mohamed (2020). "A case for new neural network smoothness constraints". In: *arXiv preprint arXiv:2012.07969*.

Ross, Andrew Slavin and Finale Doshi-Velez (2018). "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients". In: *Thirty-second AAAI conference on artificial intelligence*.

Rudner, Tim GJ, Zonghao Chen, and Yarin Gal (2020). "Rethinking function-space variational inference in Bayesian neural networks". In: *Third Symposium on Advances in Approximate Bayesian Inference*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Salimans, Tim, Andrej Karpathy, Xi Chen, and Diederik P Kingma (2017). "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications". In: *arXiv preprint arXiv:1701.05517*.

Samarth Sinha Sayna Ebrahimi, Trevor Darrell (2019). "Variational Adversarial Active Learning". In: *arXiv preprint arXiv:1904.00370*.

Saul, Lawrence K, Tommi Jaakkola, and Michael I Jordan (1996). "Mean field theory for sigmoid belief networks". In: *Journal of artificial intelligence research* 4, pp. 61–76.

Scheffer, Tobias, Christian Decomain, and Stefan Wrobel (2001). "Active hidden markov models for information extraction". In: *International Symposium on Intelligent Data Analysis*. Springer, pp. 309–318.

Scheirer, Walter J, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult (2012). "Toward open set recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 35.7, pp. 1757–1772.

Schölkopf, Bernhard, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt (2000). "Support vector method for novelty detection". In: *Advances in neural information processing systems*, pp. 582–588.

Sener, Ozan and Silvio Savarese (2018). "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: *International Conference on Learning Representations*.

Settles, Burr (2009). "Active learning literature survey". In.

Shalit, Uri, Fredrik D Johansson, and David Sontag (2017). "Estimating individual treatment effect: generalization bounds and algorithms". In: *International Conference on Machine Learning*. PMLR, pp. 3076–3085.

Shen, Yanyao, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar (2018). "Deep Active Learning for Named Entity Recognition". In: *International Conference on Learning Representations*.

Siddhant, Aditya and Zachary C Lipton (2018). "Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study". In: *arXiv preprint arXiv:1808.05697*.

Simonyan, K. and A. Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*.

Smith, Lewis, Joost van Amersfoort, Haiwen Huang, Stephen Roberts, and Yarin Gal (2021). "Can convolutional ResNets approximately preserve input distances? A frequency analysis perspective". In: *arXiv preprint arXiv:2106.02469*.

Snell, Jake, Kevin Swersky, and Richard Zemel (2017). "Prototypical networks for few-shot learning". In: *Advances in Neural Information Processing Systems*, pp. 4077–4087.

Snoek, Jasper, Hugo Larochelle, and Ryan P Adams (2012). "Practical Bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems*, pp. 2951–2959.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Tan, Mingxing and Quoc Le (2019). "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR, pp. 6105–6114.

Theis, Lukas, Aaron van den Oord, and Matthias Bethge (2016). "A note on the evaluation of generative models". In: *International Conference on Learning Representations*, pp. 1–10.

Titsias, Michalis (2009). "Variational learning of inducing variables in sparse Gaussian processes". In: *Artificial Intelligence and Statistics*, pp. 567–574.

Tong, Simon (2001). *Active learning: theory and applications*. Vol. 1. Stanford University USA.

Tran, Dustin, Jeremiah Zhe Liu, Michael W Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda E Mariet, Huiyi Hu, Neil Band, Tim G. J. Rudner, Zachary Nado, Joost van Amersfoort, Andreas Kirsch, Rodolphe Jenatton, Nithum Thain, E. Kelly Buchanan, Kevin Patrick Murphy, D. Sculley, Yarin Gal, Zoubin Ghahramani, Jasper Snoek, and Balaji Lakshminarayanan (2022). "Plex: Towards Reliability using Pretrained Large Model Extensions". In: *Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*.

Van den Oord, Aaron, Oriol Vinyals, et al. (2017). "Neural discrete representation learning". In: *Advances in Neural Information Processing Systems*, pp. 6306–6315.

Van der Maaten, Laurens and Geoffrey Hinton (2008). "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11.

Van Amersfoort, Joost, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal (2021). "On feature collapse and deep kernel learning for single forward pass uncertainty". In: *arXiv preprint arXiv:2102.11409*.

Van Amersfoort, Joost, Lewis Smith, Yee Whye Teh, and Yarin Gal (2020). "Uncertainty Estimation Using a Single Deep Deterministic Neural Network". In: *International Conference on Machine Learning*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Wang, Keze, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin (2017). "Cost-effective active learning for deep image classification". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.12, pp. 2591–2600.

Wen, Yeming, Dustin Tran, and Jimmy Ba (2020). "Batchensemble: an alternative approach to efficient ensemble and lifelong learning". In: *arXiv preprint arXiv:2002.06715*.

Wilson, Andrew G and Pavel Izmailov (2020). "Bayesian deep learning and a probabilistic perspective of generalization". In: *Advances in neural information processing systems* 33, pp. 4697–4708.

Wilson, Andrew Gordon, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing (2016a). "Deep kernel learning". In: *Artificial Intelligence and Statistics*, pp. 370–378.

Wilson, Andrew Gordon, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing (2016b). "Stochastic variational deep kernel learning". In: *Advances in Neural Information Processing Systems*, pp. 2586–2594.

Wilson, Andrew Gordon, Pavel Izmailov, Matthew D Hoffman, Yarin Gal, Yingzhen Li, Melanie F Pradier, Sharad Vikram, Andrew Foong, Sanae Lotfi, and Sebastian Farquhar (2022). "Evaluating approximate inference in Bayesian deep learning". In: *NeurIPS 2021 Competitions and Demonstrations Track*. PMLR, pp. 113–124.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms". In: *arXiv preprint arXiv:1708.07747*.

Yang, Zhengyuan, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yumao Lu, Zicheng Liu, and Lijuan Wang (2021). "An empirical study of gpt-3 for few-shot knowledge-based vqa". In: *arXiv preprint arXiv:2109.05014*.

Yeung, Raymond W (1991). "A new outlook on Shannon's information measures". In: *IEEE transactions on information theory* 37.3, pp. 466–474.

Zagoruyko, Sergey and Nikos Komodakis (2016). "Wide Residual Networks". In: *BMVC*.

Zhang, Yao, Alexis Bellot, and Mihaela Schaar (2020). "Learning overlapping representations for the estimation of individualized treatment effects". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1005–1014.

Zintgraf, Luisa M, Leo Feng, Cong Lu, Maximilian Igl, Kristian Hartikainen, Katja Hofmann, and Shimon Whiteson (2021). "Exploration in approximate hyper-state space for meta reinforcement learning". In: *International Conference on Machine Learning*. PMLR, pp. 12991–13001.