Structure and Uncertainty in Deep Learning

Lewis Smith

Kellogg College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2022

Abstract

Designing uncertainty-aware deep learning models which are able to provide reasonable uncertainties along with their predictions has long been a goal for parts of the machine learning community. Such models are also frequently desired by practitioners. The most widespread and obvious method to provide this to this is to take existing deep architectures and attempt to apply existing Bayesian techniques to them, for instance, by treating the weights of the neural network as random variables in a Bayesian framework. This thesis attempts to answer the question: **are existing neural network architectures the *best* way to get reasonable uncertainty?** In the first part of this thesis, we present research on the uncertainty behaviour of Bayesian neural networks in an adversarial setting, which demonstrates that, while a Bayesian approach improves significantly on deterministic networks *near* the data distribution, the extrapolation behaviour is undesirable, as standard neural network architectures have a structural bias toward confident extrapolation. Motivated by this, we then explore two alternatives to standard deep learning architectures which attempt to address this issue. First, we describe a novel generative formulation of *capsule networks*, which attempt to impose structure on a learning task by making strong assumptions about the structure of scenes. We then use this generative model to examine whether these underlying assumptions are useful, arguing that they in fact have significant flaws. Second, we explore *bilipschitz models*, a family of architectures which address the more limited goal of ensuring prior reversion in deep neural networks. These are based on deep kernel learning, attempting to control the behaviour of neural networks out of distribution by using final classification layers which revert to a prior as the distance to a set of support vectors increases. To maintain this property while using a neural feature extractor, we describe a novel 'bilipschitz' regularisation scheme for these models, based on preventing feature collapse by imposing a constraint motivated by work on invertible networks. We describe various useful applications of these models, and analyse why this regularisation scheme still appears to be effective even when the original motivation behind it no longer holds, in particular, where the feature dimensionality is lower than the input.

We conclude that, while capsule networks are likely not a promising direction, the models discussed in the final part of the thesis are a fruitful area for future research, as a promising potential alternative to standard Bayesian deep learning approaches in many applications.

Structure and Uncertainty in Deep Learning

Lewis Smith

Kellogg College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2022

Acknowledgements

Personal

I would like to thank my supervisor, Yarin, for his indispensable inspiration, insight and motivation over the years, and for shaping and refining the course of my research in the field. I also thank the rest of the OATML lab, both those who I directly worked with on the work included in this thesis, whose contribution to the direction and implementation of those lines of research made them possible, and more broadly those whose friendship and discussion made my time in Oxford both more enjoyable and more intellectually stimulating. I would like to particularly thank Joost van Amersfoort, Sebastian Farquhar, Milad Alizadeh, Lisa Schut, Mizu Nishikawa-Toomey, Andrew Jesson and Mark van der Wilk for our collaborations and discussion during my DPhil.

I thank my family and friends, for their support and putting up with my complaining, particularly Ben Rout, who I was fortunate to share a house with during the pandemic lockdowns.

Finally, I would like to thank Yuhe Liu, despite everything, for her love in the years in which this thesis was written.

Institutional

This work was made possible due to funding via the AIMS CDT and the Engineering and Physical Sciences Research Council (EPSRC), grant number EP/L015897/1.

Abstract

Designing uncertainty-aware deep learning models which are able to provide reasonable uncertainties along with their predictions has long been a goal for parts of the machine learning community. Such models are also frequently desired by practitioners. The most widespread and obvious method to provide this to this is to take existing deep architectures and attempt to apply existing Bayesian techniques to them, for instance, by treating the weights of the neural network as random variables in a Bayesian framework. This thesis attempts to answer the question: **are existing neural network architectures the *best* way to get reasonable uncertainty?** In the first part of this thesis, we present research on the uncertainty behaviour of Bayesian neural networks in an adversarial setting, which demonstrates that, while a Bayesian approach improves significantly on deterministic networks *near* the data distribution, the extrapolation behaviour is undesirable, as standard neural network architectures have a structural bias toward confident extrapolation. Motivated by this, we then explore two alternatives to standard deep learning architectures which attempt to address this issue. First, we describe a novel generative formulation of *capsule networks*, which attempt to impose structure on a learning task by making strong assumptions about the structure of scenes. We then use this generative model to examine whether these underlying assumptions are useful, arguing that they in fact have significant flaws. Second, we explore *bilipschitz models*, a family of architectures which address the more limited goal of ensuring prior reversion in deep neural networks. These are based on deep kernel learning, attempting to control the behaviour of neural networks out of distribution by using final classification layers which revert to a prior as the distance to a set of support vectors increases. To maintain this property while using a neural feature extractor, we describe a novel 'bilipschitz' regularisation scheme for these models, based on preventing feature collapse by imposing a constraint motivated by work on invertible networks. We describe various useful applications of these models, and analyse why this regularisation scheme still appears to be effective even when the original motivation behind it no longer holds, in particular, where the feature dimensionality is lower than the input.

We conclude that, while capsule networks are likely not a promising direction, the models discussed in the final part of the thesis are a fruitful area for future research, as a promising potential alternative to standard Bayesian deep learning approaches in many applications.

Contents

List of Figures

In the days when Sussman was a novice, Minsky once came to him as he sat hacking at the PDP-6. "What are you doing?", asked Minsky. "I am training a randomly wired neural net to play Tic-Tac-Toe" Sussman replied. "Why is the net wired randomly?", asked Minsky. "I do not want it to have any preconceptions of how to play", Sussman said. Minsky then shut his eyes. "Why do you close your eyes?", Sussman asked his teacher. "So that the room will be empty."

At that moment, Sussman was enlightened.

— Well known AI joke, original author unknown

1 Introduction & Background

Contents

1.1 Why Uncertainty-Aware Deep Learning?

In many applications, we are interested in *uncertainty-aware* models which are able, as well as making predictions, to quantify their uncertainty in these predictions in an informative way. For any model which produces a set of scores over outcomes, such as a classification model, it is generally trivial to interpret these as a probability distribution by normalising them. But we are interested, not just in having some arbitrary probability distribution, but in how *useful* these probabilities are. There are many practical techniques which are possible if a model is able to calculate its uncertainty accurately, such as rejecting images which have been tampered with, as we explore in Chapter 2, deciding whether a model is well-supported enough by data to make medical inferences based on the predictions, as we explore in 7.3.4, or choosing which points to acquire labels about, as in active learning. In all of these scenarios, we want to change our decision rule based on uncertainty expressed by the model.

Uncertainty aware deep learning is a *goal*, rather than a specific methodology. Our goal in this thesis is primarily accurate and reliable assessments of the models uncertainty, however these are achieved. There are various criteria which could be used to decide how useful the model uncertainty is, which may depend on the application. In this thesis, the scenario most frequently considered is how a model should behave when presented with data out of the training distribution; we would ideally like our model to default to some controlled prior prediction and express high uncertainty when making predictions not well supported by the training data.[1] This addresses the common use case, outlined above, of using the model uncertainty to reject uncertain inputs or refer them to an expert. The ability, discussed above, to separate aleatoric and epistemic uncertainty, as we discuss in Section 1.5 is also useful in some applications, such as active learning (Houlsby, 2014).

The question that this theses addresses is **are existing neural network architectures the *best* way to get reasonable uncertainty in deep learning**, in applications like these? In such applications, the most conventional current approach is to apply Bayesian methods, such as variational inference over the weights, to standard deep learning architectures. However, whether this is the best way of achieving the *goal* of accurate and robust uncertainty estimation, while retaining the performance advantages of deep learning, is the question this thesis attempts to address, by exploring alternative methodologies aimed at robust uncertainty estimation.

[1]This criterion reflects the focus of the models in this thesis, which is generally supervised classification or regression. In other applications, the most relevant criteria for good uncertainty might be different; for instance, an election forecaster would probably be more interested in the calibration of their model (that is, whether the model is correct on events it assigned an 70% probability about 70% of the time)

In this introduction, we provide some important background to the main work presented. Firstly, we describe why probability theory provides a natural framework for the quantification of uncertainty, before introducing some standard arguments and methods of both deep learning and the Bayesian approach in deep learning. We also discuss the concepts of aleatoric and epistemic uncertainty, and give an overview of the work in the remainder of the thesis.

1.2 Probability & Subjectivity

If our goal is to quantify uncertainty, then we need a mathematical framework in which to express and manipulate these uncertainty scores. The natural mathematical framework for reasoning over uncertainty is probability theory, especially the Bayesian interpretation of probability, where the probability of an event is simply a quantified expression of belief, subject to some logical constraints.

Are probabilities just beliefs? We might, alternatively, want to relate probabilities to real, observable proportions of repeated experiments; saying that 'x occurs with probability 0.4' informally means 'if we repeat the experiment enough times, the proportion of trials in which x occurs will be about 40%'.[2]

However, we frequently want to talk about probabilities of events where the notion of a repeated experiment doesn't really make any sense; for example 'what is the probability that candidate A will win the next election' is a something that seems natural to think about. But there is no realistic sense in which this event can ever be a repeated experiment; either A will win or not, and we can't re-run the same election multiple times, or observe multiple realisations of it.

Even in the most obviously prototypical random events, like flipping a coin, there is an implicit assumption about how precisely the experiment of flipping the coin is being repeated. The notion that if we flip a coin a large number of times, the ratio of heads to tails will be equal seems obvious. But given enough information, the coin flip, is a deterministic function of the initial conditions of the

[2]Expectation is an intuitive notion, but giving it a truly rigorous mathematical definition in terms of limits is fairly technical (see, for instance, Pollard (2002)). Doing so is not particularly important for the work discussed in this thesis

coins trajectory, and if we replicated the flips carefully enough, for instance, by constructing a specialised machine to launch the coin with a very precise velocity, we could get the same result every time (Diaconis et al., 2007). So the randomness here is really a matter of *perspective*, implying a model for how precisely we are able to observe something [3] The apparent non-determinism in the coin flip comes from our *uncertainty* over the initial conditions of the coin, and the sensitivity of its dynamics to these initial conditions.

Therefore, the 'Bayesian'[4] perspective understands probability as simply an expression of subjective belief, rather than as any objective statement about the outcome of theoretically repeated experiments. When we say 'a coin flip has a 50% chance of coming up heads' we are simply attempting to quantify the degree to which we find these statements plausible; heads and tails are equally believable outcomes of this process, given we know nothing else about the coin or how it is flipped.

Probability theory, in this interpretation, is an extension of *logic* from binary truth or falsehood to a system where degrees of belief can be quantified by real numbers. Assigning a probability to a statement, say the probability that it will rain tomorrow, is simply assigning a numerical value to our degree of belief in that proposition. It's obvious why a numerical calculus for degrees of belief is the natural mathematical framework for manipulating uncertainty.

Probabilities may be subjective, but they are not totally arbitrary, of course: a probability distribution is a *logically consistent* set of beliefs over a set of propositions. Given a particular model of the world, our beliefs about particular propositions may constrain our beliefs about other propositions, following the mathematical machinery of probability theory.

---

[3]This is almost certainly true of essentially any physical system at the scales we are likely to care about. At the level of quantum phenomena there do seem, to the best of current knowledge, to be genuinely irreducibly random processes. But at the 'human scale' problems where we largely care about applying the techniques discussed here, mechanics can be assumed to be classical to a very good approximation

[4]While Bayes himself was the first to use a version of what is now known as Bayes' theorem to solve a statistical problem, the subjective perspective on probability theory that bears his name is more accurately credited to Laplace.

It can be shown (Cox, 1946) that the standard machinery of probability can be derived a set of reasonable specifications for a calculus of degrees of belief: we can obtain probability theory assuming only that 'degrees of belief' are representable by real numbers, and the some conditions for logical consistency across these beliefs. One way to state these consistency requirements are (Jaynes, 2003)

1. If a degree of belief can be derived in multiple ways, they must all lead to the same answer.

2. A derivation of a degree of belief must use all relevant information presented to an agent; it is not permissible to arbitrarily ignore information.

3. Equivalent states of knowledge must lead to equivalent degrees of belief (e.g there can be no dependence on the arbitrarily labelling of certain states)

In the opinion of the author the desirability of using reasoning patterns which obey these rules above when designing autonomous decision-making machines is fairly self evident[5].

In summary, then, a 'probability distribution' is simply a more-or-less arbitrary expression of our subjective beliefs. The rules of probability require that these beliefs, and subsequent beliefs derived from updating them, are logically coherent, but there

[5]There are some philosophical objections to the argument that Cox type axioms lead inevitably to standard probability theory. The most difficult are centred around the problem of justifying why we should accept these axioms; are there situations where it would make sense for a rational agent to have non-consistent beliefs? Is there an operational cost to non-consistency? Classically, one can construct 'Dutch Book' type arguments, demonstrating that an agent who is willing both to express inconsistent beliefs and act on them by taking risk-neutral bets at the odds implied by these beliefs is vulnerable to having a series of bets made against them which guarantee a loss (De Finetti, 1937). In this case, maintaining their non-consistent beliefs imposes a quantifiable potential cost on the agent. However, such arguments are not watertight - what if the agent is not risk neutral? What if an agent refuses to bet? etc. This is complicated further by the fact that it is generally accepted that *people* are not rational actors in this sense; for instance, there is considerable empirical evidence that people violate several of these axioms in practice; decisions made by people depend on, for example, on the order in which options are presented to them (Kahneman and Tversky, 1979). There is also the issue of practicality; perhaps a rational agent will know that obeying these axioms would be desirable, but too difficult or expensive to apply, and so could rationally decide to rely on a cheaper irrational heuristic instead of an expensive system which obeys these axioms. In practice, of course, even 'Bayesian' machine learning almost invariably relies on approximations to the ideal calculation we would get by applying Bayes' rule. However, a rigorous defence of the philosophy of subjectivist probability is outside the scope of this thesis

is no real restriction as to what it makes sense to talk about using probability. Using these rules to update a set of beliefs in order to make decisions based on observed data is the central business of the statistical approach to artificial intelligence.

One of the rules of probability which arises from these axioms is the *chain rule* that for two events, A and B

$$P(A \cup B) = P(A \mid B)P(B) = P(B \mid A)P(A) \tag{1.1}$$

Intuitively, this is a consequence of the requirement that degrees of belief must be consistent with one another; we must have the same belief in $A \cup B$ regardless of whether we first see observe B, then A, or first observe A, followed by B. From this, via simple re-arranging, we can derive *Bayes theorem*

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{1.2}$$

This has direct application to making data driven decisions; if we have a hypothesis H and a model for the probability of seeing some observations O, given H, we want to know how to update our hypothesis based on this new information. Bayes rule gives us a simple update for the *posterior* probability that H is true from our *prior belief* $P(H)$;

$$P(H \mid O) = \frac{P(O \mid H)P(H)}{P(O)} \tag{1.3}$$

This is in many ways the probabilistic analogue of logical deduction. The advantage of a probabilistic approach is that it gives us a single, rigorous framework for dealing with uncertainty, allowing us to propagate our uncertainty from a prior belief to some posterior deduction about the state of the world. In order to improve the uncertainty expressed by models, trying to apply this Bayesian treatment to their unknown parameters is therefore a natural approach, though ultimately we will decide the best way to express uncertainty empirically.

1.2.1 Probabilistic machine learning

Machine learning is an approach to the design of a rule or procedure for automatic decision making. Rather than specifying automatic rules by hand, machine learning attempts to derive decision rules at least in part from observed data. The loose analogy is that the system is 'learning' from 'observation' in a mechanical way, hence the name.[6] For instance, in the classical setup of 'supervised' learning, we have a dataset $\mathcal{D} = \{x_i, y_i\}$, and the goal is to predict the outputs y_i from the corresponding inputs x_i with some automatic rule – a function – mapping the inputs to the outputs. We also generally assume some cost function or 'loss' which measures how good our automatic rule is, such as the accuracy or the mean squared error of the model predictions. We can then choose the model which, for instance, has the lowest loss on a held-out set of data, having used the rest to choose the parameters of the model by optimising for the loss on a 'training' set. In probabilistic terms, this is equivalent to choosing a model $P(Y|X, \theta)$, where θ are some unknown model parameters indexing the family, together with some prior probability $p(\theta)$, and then trying to infer the posterior distribution $P(\theta|\mathcal{D}) \propto P(\mathcal{D} \mid \theta)P(\theta)$ using Bayes' theorem.

Using the ideal Bayesian approach is, for all but a few special cases, computationally prohibitive. Both calculating the normalising constant of the posterior $P(\theta \mid \mathcal{D})$ and computing predictions by marginalising over θ to compute $P(x \mid y) = \int P(x \mid y, \theta)P(\theta \mid \mathcal{D})d\theta$ involve an integral which, for many interesting models, is mathematically intractable and very high dimensional, making direct numerical approximation to these expressions difficult. Rather than restrict ourselves to models where these integrals can be carried analytically, the probabilistic machine learning field has devoted a great deal of effort into improving the approximation of these quantities for existing classes of model, by developing various forms of tractable approximations to ideal Bayesian inference.

[6]There is a clear gap between the meaning of 'learning' in this sense and the use of the word in colloquial English, but this term is ubiquitous in practice.

1.2.2 Deep learning

Most of the work in this thesis is concerned with so called *deep* models. Deep learning is conceptually fairly simple: we have a parameterised function $f(x, \theta) \to y$ mapping from inputs x to outputs y. This function is implemented in a *differentiable programming* framework, such that the gradient of any scalar function of the outputs with respect to the parameters θ can be efficiently obtained using backward-mode automatic differentiation (Baydin et al., 2018). We can then fit the parameters θ by minimising a loss function on the outputs of the network, using gradient based optimisation.

Arguably, and with the benefit of considerable hindsight, the main difference between deep learning and classical machine learning is the use of generic and scalable approaches which can be applied to very large classes of model architecture rather than specialised models with corresponding efficient learning algorithms. In the pre-deep learning era, computational and data constraints led made generic approaches impractical, and much energy was focused instead on models, such as Gaussian mixtures or support vector machines, which permitted the finding of solutions for θ using specialised algorithms. In contrast, deep learning allows the more-or-less arbitrary design of a function via composition of differentiable 'blocks', or simple functions[7] Since the result is differentiable as well, we can then use generic gradient descent algorithms to fit the parameters of the model. While this has few guarantees that it will converge to an optimal solution in most models of interest, it frequently turns out that poor optimisation in a good model is better than good optimisation in a poor model.

While the differentiable programming framework gives a great deal of flexibility in *theory* to the kind of models which can be learned in this way, in practice gradient descent is not an practical way to optimise a truly arbitrary program. Neural network design tends to use few fundamental 'building blocks', often called

[7]Hence a 'deep' stack of nonlinear blocks, as opposed to a 'shallow' model like logistic regression or kernel machines which act directly on the input features, or on a single nonlinear expansion of the features.

'layers' which are known to have reasonable numerical properties when fit using gradient descent, which can then be combined.

Of these, the most important are:

- **Linear Layers** These are a general matrix mapping: $f(x) = Ax + b$, where A, b are learnable matrices.
- **Convolution layers** $f(x) = K * x + b$, where K is a learnable convolution kernel and b a bias. Convolution is also a linear mapping, but it is an extremely useful one because of the massively restricted number of learnable parameters. In addition, convolution is an extremely important operation since convolutions are *equivariant* to translations. More general convolutions can be defined which are invariant to other Lie groups (Weiler and Cesa, 2019). Convolutions are an extremely important feature of neural networks designed to process data which lies on a multidimensional grid, such as time series in one dimension, 2D images or 3D voxels.
- **Nonlinearities** These are simple nonlinear operations, generally applied pointwise on vectors. Many of the learnable parameter blocks in neural networks are linear, and since compositions of linear functions are themselves linear, nonlinearities are essential for a stack of such layers to have increased expressive power. The most common nonlinearity used today is the $ReLU$, the function $f(x) = \max(0, x)$.
- **Normalisation Layers** These are layers which are designed to approximately normalise the input of another layer to be zero mean and unit standard deviation. The most notable is BatchNorm (Ioffe and Szegedy, 2015), which stores an exponential moving average of the mean and standard deviation of its inputs and uses these to scale its input.
- **Attention** These can be thought of as a weighted aggregation function between corresponding 'value' vectors v_j, corresponding 'key' vectors k_j, and a set of 'query' vectors q_i of the same dimension as the keys. A similarity

function is used to produce a weight between each query q_i and key k_j (usually the dot product), then the values are aggregated with a weighted average, using the softmax over the similarity weights, producing an aggregated value for each query. Self attention, where each element of a sequence is used as both a value and a query, is a key component of the *transformer* introduced in Vaswani et al. (2017), which has recently become increasingly important, especially in natural language processing.

- **Recurrent Layers** These take as input both an input x and a 'hidden state' z, and are intended to be applied in a sequential fashion, with the hidden state propagated, so that, applied to a sequence $x_1, x_2, \ldots$, the network produces the output $z_1, z_2, \ldots$, with $z_i = f(x_i, z_{i-1})$. While this is more of a large-scale architecture choice than a building block, in practice specialised architectures like the LSTM (Hochreiter and Schmidhuber, 1997b) are necessary for training to be stable in these models. While RNNs are important in some applications, we do not really discuss them in the work in this thesis.

- **Residual Connections** Where g is a composition of the above blocks, a residual connection is simply adding g to the identity; $f(x) = x + g(x)$. While this seems a simple idea, in practice it is extremely important for optimisation (He et al., 2015a; Hochreiter and Schmidhuber, 1997a), as the residual connections keep the Lipschitz constant of the block at initialisation close to one, which is an important requirement for the stability of gradient based training (Hayou et al., 2019). This stabilisation of the gradient allowed the construction of much deeper models (He et al., 2015b). The residual connection plays a very important role in the models discussed in Part III of this thesis.

1.3 Bayesian Deep Learning

In conventional deep learning, we treat the unknown parameters of the deep model as deterministic parameters to be determined by optimisation of a loss function. This is contrary to the Bayesian philosophy, which says we should treat unknown parameters of our model as *random*, rather than assuming we can know them with complete certainty. Especially in an expressive model like a deep net, many settings of the parameters are likely to give a reasonable loss on the available data. An obvious question, therefore, is whether we can make these models 'Bayesian', by integrating uncertainty over the unknown model parameters of a deep model, rather than treating these as fixed parameters to be learnt. A more Bayesian approach would be to attempt to directly model the posterior over the network weights $P(w \mid D)$, then average over this distribution when making predictions with the model.

The posterior distribution over the weight space is highly intractable due to the large dimensionality and nonlinearity of neural architectures. The two main approaches to approximate this are to attempt to sample directly from the posterior using, for example, Markov Chain Monte Carlo methods (Neal, 1995), or to attempt to learn a simpler approximation to the posterior using *variational inference*, or, more rarely, other methods like the Laplace approximation (MacKay, 1992). Much research on Bayesian machine learning can be summarised as developing better ways to scale or simplify these two basic approaches.

Markov Chain Monte Carlo methods are an extremely useful tool, and remain the 'gold standard' for obtaining an approximation to the true Bayesian posterior due to their strong theoretical guarantees. However, they are also extremely expensive compared to standard deep learning, and so are not really a practical replacement for deep models in applications, though they remain an important theoretical tool for researchers to use to explore the properties of ideal Bayesian behaviour. As a result, in this thesis we focus far more on variational inference, as this can have computational cost much more comparable to standard supervised learning.

1.4 Variational Inference

Variational inference, as mentioned, is an important technique in general Bayesian machine learning, and it is used particularly heavily in this thesis, so it is worth introducing it in more detail. Variational inference refers to a family of methods for approximating an intractable target distribution $p(x)$, with a simpler approximating distribution $q(x)$, described by sufficient statistics ϕ and written $q(x;\phi)$. Variational inference assumes that it is relatively cheap to evaluate an function $p^*(x)$ which is proportional to p, but that sampling from p directly or computing the normalising constant $Z = \int p^*(x)dx$ is impossible. For example, in the case of deep learning, we wish to approximate the posterior $p(\theta \mid \mathcal{D})$ of a parameterised model. Using Bayes rule, we have that $p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)p(\theta)$. This is easy to evaluate; $p(\theta)$ is a prior over weights, which is normally chosen to be a simple to evaluate, and $p(\theta \mid \mathcal{D}) = \sum_i p(y_i \mid x_i, \theta)$ is simply the likelihood of the data, and is similarly easy to evaluate. However, computing the normalising constant of this distribution requires taking an integral over the whole weight space, which is both high dimensional and nonlinear.

This variational distribution q (or equivalently, its parameters ϕ) is chosen by minimising the (reversed) *Kullbeck-Liebler divergence*, or relative entropy, between the target and the variational distribution, that is, minimising

$$D_{KL}\left(q : p\right) = \int q(x) \log\left(\frac{q(x)}{p(x)}\right) dx \tag{1.4}$$

Substituting in $p(x) = p^*(x)/Z$, this can be re-arranged to obtain

$$\begin{aligned}
D_{KL}\left(q : p\right) &= \int q(x) \log\left(\frac{Zq(x)}{p^*(x)}\right) dx && (1.5)\\
&= \log Z + \int q(x) \log\left(\frac{q(x)}{p^*(x)}\right) dx && (1.6)\\
\implies \log Z &= D_{KL}\left(q : p\right) + \int q(x) \log\left(\frac{p^*(x)}{q(x)}\right) dx && (1.7)\\
\implies \log Z &\geq \int q(x) \log\left(\frac{p^*(x)}{q(x)}\right) dx && (1.8)
\end{aligned}$$

since the KL divergence is strictly positive. The term $\mathcal{L} = \int q(x) \log\left(\frac{p^*(x)}{q(x)}\right) dx$ is therefore a *lower bound* on the log of the normalising constant Z. Since this term takes the form of an expectation over q, it is tractable to evaluate it by sampling from q and using a Monte-Carlo approximation. In the Bayesian case, as mentioned, the normalising constant Z is equal to the model evidence[8], and so $\mathcal{L}$ is called the *evidence lower bound* or ELBO. Z is independent of the variational distribution q, and so we have no need to evaluate it in order to chose a q to maximise $\mathcal{L}$, which is the major advantage of variational inference.

In some cases, the optimal variational distribution from a given family can be found analytically using the calculus of variations (Mackay, 2003), or can be solved analytically conditioned on some partitioning of the latent variables, as in the classical algorithm for Gaussian mixture models (Bishop, 2006)[9], leading to a block co-ordinate ascent algorithm for optimising the ELBO.

From a deep learning perspective, however, the big advantage of variational inference is that it translates the problem of Bayesian inference, which is chiefly concerned with intractable integrals, into a problem of choosing parameters ψ which maximise an objective function. This can be approached using the standard toolbox of automatic differentiation, combined with generic gradient descent algorithms, meaning that this approach is fairly compatible with various deep learning methods. q can be chosen to be more or less any distribution, though in practice the choice of q will have significant effects on both how easy it is to learn the optimal q, and on the quality of inference, as even the optimal q might not provide a particularly good approximation to the posterior.

1.5 Epistemic & Aleatoric Uncertainty

One distinction which is sometimes useful when attempting to quantify uncertainty is to distinguish between *epistemic* and *aleatoric* uncertainty. *Epistemic* uncertainty is uncertainty in our predictions which comes from our lack of data. In particular, in Bayesian machine learning, this is uncertainty which comes from our uncertainty

[8]The normalising constant $p(O \mid H)$ of the observations given the hypothesis is often referred to as the model evidence in Bayesian modelling, as it is used to calculate the relative plausibility of two different hypotheses H_1 and H_2. So $p(O \mid H_1)$ can be thought of as the evidence for H_1, given the data

[9]For a Gaussian mixture model, the optimal variational distribution is not tractable, but optimal distributions for the positions of the cluster means conditioned on cluster assignments and for cluster assignments conditioned on cluster means are both derivable, leading to an EM type algorithm

about the parameters of an unknown model. *Aleatoric* uncertainty is uncertainty which is modelled as irreducible randomness in the data. For instance, in a classical regression model, where we assume that the data is generated by a linear process $y \sim \mathcal{N}(\theta x, \sigma)$, then aleatoric uncertainty comes from our uncertainty about the regression parameter θ, whereas other noise is modelled as observation noise from the normal distribution around this mean prediction.

As this example should make clear, these distinctions are a part of the modelling choice we make when we set up a problem: that is, whether a given source of noise is considered epistemic or aleatoric uncertainty is not an objective fact about the data but dependent on the data and model we are using. As already discussed, even the most canonical example of aleatoric noise, the coin flip can be made deterministic given enough information about the initial conditions, so if we were able to observe these in sufficient detail this could become epistemic noise, though a simpler model might assume it was random because this granularity is not available. Similarly, a linear model will be forced to model any non-linear trends in the data as noise (aleatoric uncertainty) while a more powerful model may be able to model these, translating uncertainty about the true trend into epistemic uncertainty. So while these are useful distinctions in the context of examining what our model is telling us, we should be careful not to take them as model independent statements.

Nevertheless, these distinctions can be useful in practice. In particular, they can be useful when we wish to make decisions based on the *a posteriori* uncertainty a particular model is expressing about a situation, as our desired actions may be different if our model is expressing aleatoric or epistemic uncertainty.

1.6 Thesis Overview

We address the central question of this thesis – **are standard Bayesian approaches the best way to get reasonable uncertainty in deep learning?** by first outlining some research conducted using standard Bayesian neural networks, as applied to deep learning, in the first part of the thesis. We investigate how the standard Bayesian deep learning toolbox behaves under adversarial attack, and outline some hypotheses for this behaviour. These hypotheses lead to outlining two potentially important properties for models which could have better uncertainty behaviour than standard Bayesian neural networks ; *invariance* and *equivariance* to symmetries present in the underlying dataset, and *prior reversion* out of distribution. In the second part of this thesis, we explore alternative architectures which attempt to realise these properties, in various ways.

Firstly, we explore *capsule networks*, which attempt to exploit the decomposition of the world into objects, and the invariance of the geometric relationships relating parts of objects to the whole. This model is of interest since this attempt to explicitly reflect the structure of scenes could potentially be a way to ensure stronger generalisation out of distribution, addressing the undesirable out-of-distribution behaviour we observe in standard structured models in Bayesian deep learning. A few variations of capsule models exist in the literature, and we attempt to unify the stated assumptions underlying these models in a probabilistic framework, as this should allow us to control behaviour out of distribution not covered by the invariances of the model by incorporating the uncertainty in our latent variables. Using our model, we explore whether capsules are a promising alternative to deep learning. We argue that our formulation reveals that there are significant flaws with the arguments motivating capsules, and that unfortunately they are unlikely to be a particularly promising alternative to standard Bayesian networks for practical applications.

As an alternative, we then explore *bilipschitz models*, a family of architectures which address the more limited goal of ensuring prior reversion in deep neural networks. These are based on similar ideas to deep kernel learning, attempting to control the behaviour of neural networks out of distribution by using final classification layers which revert to a prior as the distance to a set of support vectors increases. This concept is not novel, but we introduce a novel regularisation scheme which attempts to control the behaviour of the neural feature extractor in order to preserve the prior-reversion of the final layer by limiting the distortion of distances. We discuss how such a model can be constructed, demonstrate some promising empirical results, and discuss the motivation and actual mechanisms behind the regularisation scheme. Unlike capsules, we feel that this kind of model represents at least a plausible alternative to standard neural network architectures for getting well behaved uncertainty in applications.

We conclude by discussing the implications of the work, and outlining possible directions for future research.

1.7 A Note on Authorship

Some of the work in this thesis is based on collaborative work, some of which has been published previously. Other than my supervisor, Yarin Gal, who was an important collaborator on all the work presented, where relevant I have included authorship statements at the start of chapters which are based on work done in collaboration with others, outlining their contributions. Unless otherwise stated, all work included is substantially my own.

1.8 Overview of Work not Included in the Thesis

During my studies, I was involved with several projects which I felt were not relevant enough to the central question addressed by this thesis, or to which I felt my contribution was not significant enough to defend the whole work as substantially my own. Many of these are in applications, particularly scientific ones, and some lead to publications. In order to make clear when I am referring to my own published work, my own name is cited in **bold** throughout in references to my own papers. I provide a brief summary below;

1. "Galaxy Zoo: Probabilistic Morphology through Bayesian CNNs and Active Learning", Walmsley, **Smith**, Lintott, Gal, et al., 2019, *Monthly Notices of the Royal Astronomical Society.* This paper describes using Bayesian CNNs, with dropout, to perform active learning in the Galaxy Zoo, a citizen science project for crowdsourcing galaxy morphology classifications from volunteers. A prototype system for active learning was investigated, with the aim of increasing the efficiency of the overall system by using the volunteer time on more difficult galaxies, while classifying more difficult examples automatically. The paper demonstrated a clear advantage of the Bayesian system over a standard neural network in terms of both active learning efficiency and the calibration of network uncertainty. The implementation and engineering for this project was carried out by Mike Walmsley, the lead author, but I had a substantial contribution to the project in terms of helping with the theory

and broad design of experiments and model architecture. In particular, the data in the galaxy zoo project has an unusual form; for each image, we have a set of volunteer responses, which can be viewed as realisations of a categorical random variable. We therefore modelled this data as a multinomial, assuming that the volunteers were indistinguishable, and so the neural network was approximating a mapping from an image to the parameters of this multinomial, i.e the probability that a randomly chosen member of the volunteer ensemble would give a particular answer to a particular prompt. We showed that using this likelihood, rather than the simple Gaussian regression of the mean response, as used in prior work, significantly improved the performance and calibration of the model. In many ways, this project is aligned with the overall themes of the thesis, as it is an application of uncertainty in deep learning. However, it uses fairly conventional approaches towards uncertainity in deep learning, with the novelty coming in the adaption of this standard scheme to the specific problem at hand, and while I made an important contribution to this project my contribution was decidedly secondary to that of the main author, Mike Walmsley, so I decided to leave it out.

2. "Towards global flood mapping onboard low cost satellites with machine learning", Mateo-Garcia, Veitch-Michaelis, **Smith**, Oprea, et al., 2021, *Scientific reports*. This paper describes a project I worked on during a placement with the Frontier Development Lab, in collaboration with the European Space Agency. The aim of this project was to develop a prototype classification system for onboard segmentation of images of flooding. The long term goal of this project was to demonstrate the feasibility of performing onboard segmentation for low-cost satellites, called cubesats. The motivation for this is that a major design constraint for such small satellites is battery life, and a major power drain is the cost of communicating with stations on the ground. Segmenting an image on board allows the satellite to reduce the amount of data that needs to be downlinked by a considerable factor. This project was successful, leading to a publication and ongoing work to deploy the system in

orbit as a proof of concept, led by other collaborators on this project. However, uncertainty was not particularly important in this application, and so it does not really fit in with the overall question this thesis attempts to attack.

3. "Liberty or depth: Deep Bayesian neural nets do not need complex weight posterior approximations", Farquhar, **Smith**, and Gal, 2020, *Advances in Neural Information Processing Systems.* This paper challenges the longstanding assumption that the mean-field approximation for variational inference in Bayesian neural networks is severely restrictive, and show this is not the case in deep networks. It proves several results indicating that deep mean-field variational weight posteriors can induce similar distributions in function-space to those induced by shallower networks with complex weight posteriors, validating our theoretical contributions empirically, both through examination of the weight posterior using Hamiltonian Monte Carlo in small models and by comparing diagonal- to structured-covariance in large settings. I was a second author on this paper, though I made substantial contributions to parts of the theory and implemented some of the experiments in the paper. This could be seen as somewhat relevant to the topic of this thesis, as a key insight of this paper is that the uncertainty of a deep network cannot be treated without also considering its structure, at least to a degree - using the same approximation in weight space produces *provably* different expressivity in shallow networks Foong et al. (2019) than in deep networks. However, I decided in the end that it was better to leave it out, as the contribution does not directly support the main argument of this thesis, and I was a secondary contributer to the paper in any case.

4. "Uncertainty quantification for virtual diagnostic of particle accelerators", Convery, **Smith**, Gal, and Hanuka, 2021, *Physical Review Accelerators and Beams.* This paper describes an application of uncertainty quantification in accelerator science. My contribution to this project was fairly modest, mainly consisting of assisting the main authors with the theory, giving feedback and

suggestions on possible uncertainty quantification methods to investigate, and suggesting neural network architectures to explore, but most of the experiments were carried out by the first author.

Part I

Drawbacks of the Weight-Space Approach

To understand how something works, figure out how to break it.

Nassim Nicolas Taleb

2 Measures of Uncertainty for Adversarial Example Detection

Contents

2.1 Introduction

Authorship Statement

This chapter is based on the conference paper “Understanding measures of uncertainty for adversarial example detection”, **Smith** and Gal, 2018, As I was first author on that paper, it is my own work.

This chapter examines some drawbacks of the weight space approach to uncertainty for standard deep neural networks. It is well known that standard neural networks are not robust: there exist small perturbations to the input of the network which produce erroneous and over-confident classification results. These perturbed inputs, known as adversarial examples (Szegedy et al., 2013), are potentially a major hurdle for the use of deep networks in safety-critical applications, or those for which security is a concern. This chapter explores whether a Bayesian approach to the weights can improve the robustness of models to adversarial attacks. The original focus of this work was on *measures* of uncertainty for adversarial example detection. While this is somewhat orthogonal to the main point of the overall thesis, we present these results and arguments as well, as they are an interesting contribution in their own right. The results presented in this chapter show that a Bayesian approach improves the robustness of a standard deterministic neural network, and can detect adversarial examples to a degree. However, our results also show that such approaches are far from perfect, and that an approximate Bayesian model still has significantly over-confident extrapolation in regions of the input space with low probability under the input manifold. We discuss possible explanations for this in more detail in the subsequent chapter.

These failings of standard neural networks, even under approximate Bayesian inference, was an important motivation for the exploration of non-standard architectures in the remainder of this thesis. While we do not address the issue of adversarial robustness in the remainder of the thesis, it is an example application which is closely connected to the broad theme of controlling the uncertainty of our model on data outside the training distribution. While in this chapter we show that idealised Bayesian inference, via HMC, resolves many of the issues with variational methods we observe, by exploring different network structures we hope to achieve similar behaviour in methods with more practical computational requirements than HMC.

One possible hypothesis for the existence of adversarial examples is that such images lie off the manifold of natural images, occupying regions where the model makes unconstrained extrapolations. Therefore, if one could calculate the distance to this manifold, this may be a reasonable metric to detect adversarial input. Of course, the 'true' distance to this manifold is impossible to know because the image manifold is not known, but we can consider proxies based on this intuition.

Hypothetically, such distances could be measured using nearest neighbour approaches, or by assessing the probability of the input under a density model on image space. However, approaches based on geometric distance are a suboptimal choice for images, as pixel-wise distance is a poor metric for perceptual similarity; similarly, density modelling is difficult to scale to the high dimensional spaces found in image recognition.

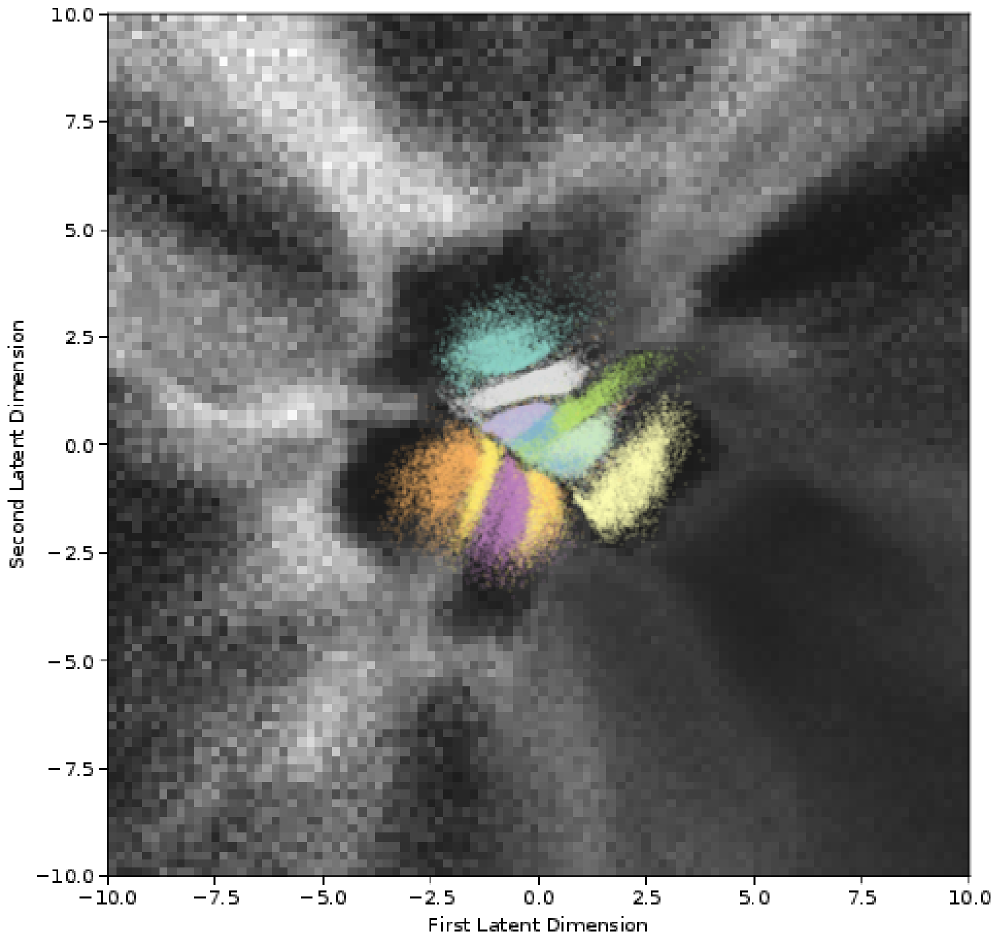


Figure 2.1: Uncertainty of a standard dropout network trained on MNIST, as measured by *mutual information*, visualised in the latent space obtained from a variational autoencoder. Colours are classes for each encoded training image. The background shows uncertainty, calculated by decoding each latent point into image space, and evaluating the mutual information between the decoded image and the model parameters. A lighter background corresponds to higher uncertainty.

Instead, we may consider proxies to the distance from the image manifold. For example, we may wish that the model uncertainty of a *discriminative* Bayesian classification model should be high for points far away from the training data, for a high-capacity model such as a deep network, as they should be able to express many functions which are all consistent with both the prior and the training data. Under the hypothesis that adversarial examples lie off the image manifold, such uncertainty could be used to identify an input as adversarial.

The uncertainty of such models is not straightforward to obtain. As discussed in the introduction, numerical methods for integrating the posterior, such as Markov Chain Monte Carlo, are difficult to scale to large datasets (Gal, 2016). As a result, approximations have been studied extensively. For example, approximate inference in Bayesian neural networks using dropout (a form of variational inference) is a computationally tractable technique (Gal and Ghahramani, 2016) which has been widely used in the literature (Gal, 2016; Leibig et al., 2017). Dropout based model uncertainty can be used for the detection of adversarial examples, with moderate success (Feinman et al., 2017; Li and Gal, 2017; Rawat et al., 2017).

However, existing research has mostly overlooked the effect of the chosen *measure for uncertainty quantification*. Many such measures of uncertainty exist, including mutual information, predictive entropy and softmax variance. (Li and Gal, 2017) for example use expected entropy, (Rawat et al., 2017) use mutual information, whereas (Feinman et al., 2017) estimate the variance of multiple draws from the predictive distribution (obtained using dropout). Further, to date, research for the identification of adversarial examples using model uncertainty has concentrated on toy problems such as MNIST, and has not been shown to extend to more realistic data distributions and larger models such as ResNet (He et al., 2015a).

In this chapter, we examine the differences between the various measures of uncertainty used for adversarial example detection. We show that the softmax variance can be seen as an approximation to the mutual information (section 2.3.2), explaining the effectiveness of this rather ad-hoc technique. We illustrate the differences between the measures by projecting the uncertainty onto lower dimensional spaces (see for example Figure 2.1).

We show that some measures of uncertainty do not distinguish between non-adversarial off-manifold images (for example image interpolations) and adversarial inputs.

Most importantly for the broader themes of this thesis, we highlight ways in which dropout fails to capture the uncertainty of the network due to its inability to capture the full uncertainty of the model. We demonstrate how this by visualising gaps in model uncertainty in the latent space (Section 2.4.2), and use this insight to propose a simple extension to dropout schemes to be studied in future research. We finish by demonstrating the effectiveness of dropout on the real-world ASIRRA (Elson et al., 2007) cats and dogs classification dataset (Section 2.4.3).

2.2 Background

2.2.1 Bayesian deep learning

In the introduction to this thesis, we gave a broad overview of the Bayesian approach to machine learning and typical deep learning architectures. Here, we explore one important method in more detail: MC Dropout (Gal and Ghahramani, 2016), which is a form of variational inference for neural networks.

2.2.2 Dropout variational inference

Variational inference is introduced in Chapter 1.4. A key design choice when using variational inference is the form of the approximating distribution q. For neural networks, a common approximating distribution is *dropout* (Srivastava et al., 2014) and its variants. In the variational framework, this means the weights are drawn from

$$\mathbf{W}_l = \mathbf{M}_l \cdot \mathrm{diag}([\mathbf{z}_{l,j}]_{j=1}^{K_l})$$

$$\text{where } \mathbf{z}_{l,j} \sim \mathrm{Bernoulli}(p_l),\ l = 1..L, j = 1..K_{l-1}$$

for a network with L layers, where the dimension of each layer is $K_i \times K_{i-1}$, and the parameters of q are $\theta = \{\mathbf{M}_l, p_l \mid l = [1..L]\}$. Informally, this corresponds to randomly setting the outputs of units in the network to zero (or zeroing the rows of the fixed matrix $\mathbf{M}_l$). Often the layer dropout probabilities p_i are chosen as constant and not varied as part of the variational framework, but it is possible to learn these parameters as well (Gal et al., 2017). Using variational inference, the expectation over the posterior can be evaluated by replacing the true posterior with the approximating distribution. The dropout distribution is still challenging to marginalise, but it is readily sampled from, so expectations can be approximated using the Monte Carlo estimator.

$$\begin{aligned}
\mathbb{E}_{p(\omega|\mathcal{D})}[f^{\omega}(x)] &= \int p(\omega|\mathcal{D}) f^{\omega}(x) \mathrm{d}\omega \\
&\simeq \int q_\theta(\omega) f^{\omega}(x) \mathrm{d}\omega \\
&\simeq \frac{1}{T} \sum_{i=1}^{T} f^{\omega_i}(x), \ \omega_{1..T} \sim q_\theta(\omega). \qquad (2.1)
\end{aligned}$$

The chief advantage of the dropout variational posterior is that, compared to other commonly used posteriors, it does not introduce as much noise into training (Farquhar et al., 2020a) and so can be readily applied to existing architectures.

2.2.3 Adversarial examples

Works by Szegedy et al. (2013) and others, demonstrating that state-of-the-art deep image classifiers can be fooled by small perturbations to input images, have initiated a great deal of interest in both understanding the reasons for why such adversarial examples occur, and devising methods to resist and detect adversarial attacks. Explanations for adversarial examples are still an active area of research, but an overview of possible causes is discussed in Chapter 3. So far, attacking has proven more successful than defence; a survey of detection methods by Carlini and Wagner (2017a) found that, with the partial exception of the method based on dropout uncertainty analysed by Feinman et al. (2017), all other investigated methods could be defeated straightforwardly.

There is no precise definition of when an example qualifies as 'adversarial'. The most common definition used is of an input $\mathbf{x}_{adv}$ which is close to a real data point $\mathbf{x}$ as measured by some L_p norm, but is classified wrongly by the network with high score. Speaking more loosely, we might define an adversarial example as a non-semantic change to an image which causes a model to predict the wrong class with high confidence.

As type of images which have troubling implications for the robustness of deep models, namely meaningless images which are nevertheless classified confidently as belonging to a particular class (see, for example, Nguyen et al., 2015). That such images can be found reveals another shortcoming of neural networks from the point of view of uncertainty, since they are far from all training data by any reasonable metric (based on either pixel-wise or perceptual distance). We refer to these as 'rubbish class examples' or 'fooling images' following Nguyen et al. (2015) and Goodfellow et al. (2014).

Many hypotheses for *why* adversarial examples exist have been proposed in the literature; these are discussed in more detail in the following chapter.

2.2.4 Measures of uncertainty

As discussed in Chapter 1.5, it is sometimes useful to distinguish between aleatoric and epistemic uncertainty in a machine learning setting.

In the classification setting, where the output of a model is a conditional probability distribution $P(y|x)$ over some discrete set of outcomes $\mathcal{Y}$, one straightforward measure of uncertainty is the entropy of the predictive distribution

$$H[P(y|x)] = -\sum_{y \in \mathcal{Y}} P(y|x) \log P(y|x). \tag{2.2}$$

However, the predictive entropy is not an entirely satisfactory measure of uncertainty, since it does not distinguish between epistemic and aleatoric uncertainties. As we will argue in Section 2.3.1, only epistemic uncertainty is relevant for adversarial examples, and aleatoric uncertainty can lead to false positives.

An attractive measure of uncertainty able to distinguish epistemic from aleatoric examples is the information gain between the model parameters and the data. Recall that the mutual information (MI) between two random variables X and Y is given by

$$\begin{aligned} I(X,Y) &= H[P(X)] - \mathbb{E}_{P(y)} H[P(X \mid Y)] \\ &= H[P(Y)] - \mathbb{E}_{P(x)} H[P(Y \mid X)]. \end{aligned}$$

The amount of information we would gain about the model parameters if we were to receive a label y for a new point x, given the dataset $\mathcal{D}$ is then given by

$$I(\omega, y \mid \mathcal{D}, x) = H[p(y \mid x, \mathcal{D})] - \mathbb{E}_{p(\omega|\mathcal{D})} H[p(y \mid x, \omega)] \tag{2.3}$$

Being uncertain about an input point x implies that if we knew the label at that point we would gain information. Conversely, if the parameters at a point are already well determined, then we would gain little information from obtaining the label. Thus, the MI is a measurement of the model's *epistemic* uncertainty.

In the form presented above, it is also readily approximated using the Bayesian interpretation of dropout. The first term we will refer to as the 'predictive entropy'; this is just the entropy of the predictive distribution, which we have already discussed. The second term is the mean of the entropy of the predictions given the parameters over the posterior distribution $p(\omega \mid \mathcal{D})$, and we thus refer to it as the expected entropy.

These quantities are not tractable analytically for deep nets, but using dropout inference and equation equation 2.1, the predictive distribution, entropy and the MI are readily approximated; (Gal, 2016):

$$\begin{aligned} p(y \mid \mathcal{D}, \mathbf{x}) &\simeq \frac{1}{T} \sum_{i=1}^{T} p(y \mid \omega_i, \mathbf{x}) && (2.4) \\ &:= p_{MC}(y \mid \mathbf{x}) \\ H[p(y \mid \mathcal{D}, \mathbf{x})] &\simeq H[p_{MC}(y \mid \mathcal{D}, \mathbf{x})] && (2.5) \\ I(\omega, y \mid \mathcal{D}, x) &\simeq H[p_{MC}(y \mid \mathcal{D}, \mathbf{x})] && (2.6) \\ &\quad - \frac{1}{T} \sum_{i=1}^{T} H[p(y \mid \omega_i, \mathbf{x})] && (2.7) \end{aligned}$$

where $\omega_i \sim q(\omega \mid \mathcal{D})$ are samples from the dropout distribution.

Other, measures of uncertainty include the empirical variance of the softmax probabilities $p(y = c \mid \omega_i, \mathbf{x})$ (with the variance calculated over i), and variation ratios (Gal, 2016), with the former commonly used in previous research on adversarial examples.

2.3 Uncertainty for Adversarial Example Detection

We start by explaining the type of uncertainty relevant for adversarial example detection under the hypothesis that adversarial images lie off the manifold of natural images, occupying regions where the model makes unconstrained extrapolations. We continue by relating the *softmax variance* measure of uncertainty to mutual information.

2.3.1 What kind of uncertainty?

Both the MI and predictive entropy should increase on inputs which lie far from the image manifold. If adversarial examples lie off the image manifold, we expect both to be effective in highlighting such inputs. However, predictive entropy could also be high *near* the image manifold, on inputs which have inherent ambiguity. Such inputs could be ambiguous images, such as an image that contains both a cat and a dog, or interpolations between classes, such as a digit that could be either a 1 or a 7. Such inputs would have high predictive probability for more than one class even in the limit of infinite data, yielding high predictive entropy (but low MI, for an expressive model). Such inputs are clearly not adversarial, but would falsely trigger a hypothetical automatic detection system[1]. We demonstrate this experimentally in the next section.

Algorithms to find adversarial examples seek to create an example image with a different class to the original, typically by either minimising the predicted probability

[1]We speculate that previous research using predictive entropy has not encountered this phenomenon due to insufficient coverage of the test cases.

of the current class for an untargeted attack, or maximising the predicted probability of a target class. This has the side-effect of minimising the entropy of the predictions, a simple consequence of the normalisation of the probability. It is interesting to highlight that this also affects the uncertainty as measured by MI; since both the mutual information and entropy are strictly positive, the mutual information is bounded above by the predictive entropy (see equation 2.3). Therefore, the model giving low entropy predictions at a point is a sufficient condition for the mutual information to be low as well. Equally, the mutual information bounds the entropy from below; it is not possible for a model to give low entropy predictions when the MI is high. It is important to realise that this means that adversarial example algorithms implicitly seek low uncertainty examples: detecting adversarial examples, at least via model uncertainty, is *not* independent of being able to fool the model without explicit detection methods. Therefore, while choosing the correct measure of uncertainty is important, it is fundamentally unable to overcome a misspecified model.

2.3.2 Mutual information and softmax variance

Some works in the literature estimate the epistemic uncertainty of a dropout model using the estimated variance of the MC samples, rather than the mutual information (Carlini and Wagner, 2017a; Feinman et al., 2017; Leibig et al., 2017). This is somewhat arbitrary for classification, but seems to work fairly well in practice. We suggest a possible explanation of the effectiveness of this measure, arguing that the softmax variance can be seen as a proxy to the mutual information.

One way to see the relation between the two measures of uncertainty is to observe that the variance is the leading term in the series expansion of the mutual information. For brevity, we denote the sampled distributions $p(y \mid \omega_i, \mathbf{x})$ as p_i and the mean predictive distribution $p_{MC}(y \mid \mathbf{x})$ as $\hat{p}$. These are in general distribution over C classes, and we denote the probability of the j^{th} class as $\hat{p}_j$ and p_{ij} for the mean and i^{th} sampled distribution respectively. The variance score is the mean variance across the classes

$$\begin{aligned}\hat{\sigma}^2 &= \frac{1}{C}\sum_{j=1}^{C}\frac{1}{T}\sum_{i=1}^{T}(p_{ij}-\hat{p}_j)^2 \\ &= \frac{1}{C}\left(\sum_{j=1}^{C}\left(\frac{1}{T}\sum_{i=1}^{T}p_{ij}^2\right)-\hat{p}_j^2\right)\end{aligned} \tag{2.8}$$

And the mutual information score is

$$\begin{aligned}\hat{I} &= H(\hat{p}) - \frac{1}{T}\sum_i H(p_i) \\ &= \sum_j \left(\frac{1}{T}\sum_i p_{ij}\log p_{ij}\right) - \hat{p}_j \log \hat{p}_j\end{aligned}$$

Using a Taylor expansion of the logarithm,

$$\begin{aligned}\hat{I} &= \sum_j \left(\frac{1}{T}\sum_i p_{ij}(p_{ij}-1)\right) - \hat{p}_j(\hat{p}_j-1) + ... \\ &= \sum_j \left(\frac{1}{T}\sum_i p_{ij}^2\right) - \hat{p}_j^2 - \left(\frac{1}{T}\sum_i p_{ij}\right) + \hat{p}_j + ... \\ &= \sum_j^{C}\left(\frac{1}{T}\sum_i^{T}p_{ij}^2\right) - \hat{p}_j^2 + ...\end{aligned} \tag{2.9}$$

we see that the first term in the series is identical up to a multiplicative constant to the mean variance of the samples. Clearly, if we are using a threshold on the uncertainty measure as a decision rule, this scaling of the metric makes no difference.

This relation between the softmax variance and the mutual information measure could explain the effectiveness of the variance in detecting adversarial examples encountered by (Feinman et al., 2017). MI increases on images far from the image manifold and not on image interpolations (on which the predictive variance increases as well), with the variance following similar trends. It makes sense that these measures are related, as intuitively the variance clearly captures disagreement between models in the Bayesian ensemble.

2.4 Empirical Evaluation

In the next section we demonstrate the effectiveness of various measures of uncertainty as proxies to distance from the image manifold. We demonstrate the difference in behaviour between the predictive entropy and mutual information on image interpolations, for interpolations in the latent space as well as interpolations in image space. We continue by visualising the various measures of uncertainty, highlighting the differences discussed above. This is further developed by highlighting shortcomings with current approaches for uncertainty estimation, to which we suggest initial ideas on how to overcome and suggest new ideas for attacks (to be explored further in future research). We finish by assessing the ideas discussed in this chapter on a real world dataset of cats vs dogs image classification.

2.4.1 Uncertainty on interpolations

We start by assessing the behaviour of the measures of uncertainty on image interpolations, comparing interpolations via convex combination ($\lambda x_1 + (1-\lambda)x_2, \lambda \in [0,1], x_i \in \mathcal{D}$) in latent space to those in image space. A convex combination in image space will clearly produce off manifold images, while we assume that moving in latent space approximates the manifold of the data fairly closely. That model uncertainty can capture what we want in practice is demonstrated in Figures 2.2 and 2.3. We see that the MI distinguishes between these on-manifold and off-manifold images, whereas the entropy fails to do so. This is necessary for the hypothesis proposed in the introduction; if we are able to accurately capture the MI, it would serve well as a proxy for whether an images belongs to the learned manifold or not.

2.4.2 Visualisation in latent space

We wish to gain intuition into how the different measures of uncertainty behave. In order to do so, we use a variational autoencoder (Kingma and Welling, 2013) to compress the MNIST latent space. We choose a latent space of two dimensions so we can use this to visualise the dataset. By decoding the image that corresponds to a point in latent space, we can classify the decoded image and evaluate the network uncertainty, thus providing a two dimensional map of the input space. Figure 2.1 shows the latent space with the uncertainty measured using the MI, calculated using dropout. Similarly, figure 2.4 shows the predictive entropy. Note the differences in

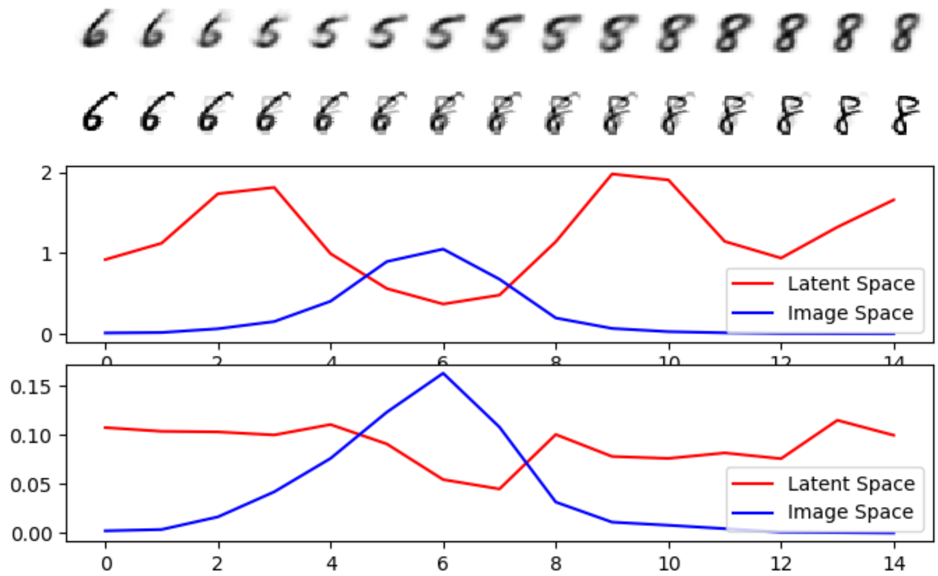


Figure 2.2: Entropy (middle) and the MI (bottom) vary along a convex interpolation between two images in latent space and image space (top). The entropy is high for regions along both interpolations, wherever the class of the image is ambiguous. In contrast, the MI is roughly constant along the interpolation in latent space, since these images have aleatoric uncertainty (they are ambiguous), but the model has seen data that resembles them. On the other hand, the MI has a clear peak as the pixel space interpolation produces out-of-sample images between the classes

uncertainty near the class cluster boundaries (corresponding to image interpolations) – the MI has low uncertainty in these regions, whereas the predictive entropy is high.

Another question of interest in this context is how well the dropout approximation captures uncertainty. The approximating distribution is fairly crude, and variational inference schemes are known to underestimate the uncertainty of the posterior, tending to fit an approximation to a local mode rather than capturing the full posterior[2].

As seen from the figures, the network does a reasonable job of capturing uncertainty close to the data. However, the network's uncertainty has 'holes'– regions where the predictions of the model are very confident, despite the images generated by the decoder here being essentially nonsense (see Figure 2.5). This suggests that, while the uncertainty estimates generated by MC dropout are useful,

[2]There are two reasons for this behaviour: firstly, that the approximating distribution q may not have sufficient capacity to represent the full posterior, and secondly, the asymmetry of the KL divergence, which penalises q placing probability mass where the support of p is small far more heavily than the reverse.

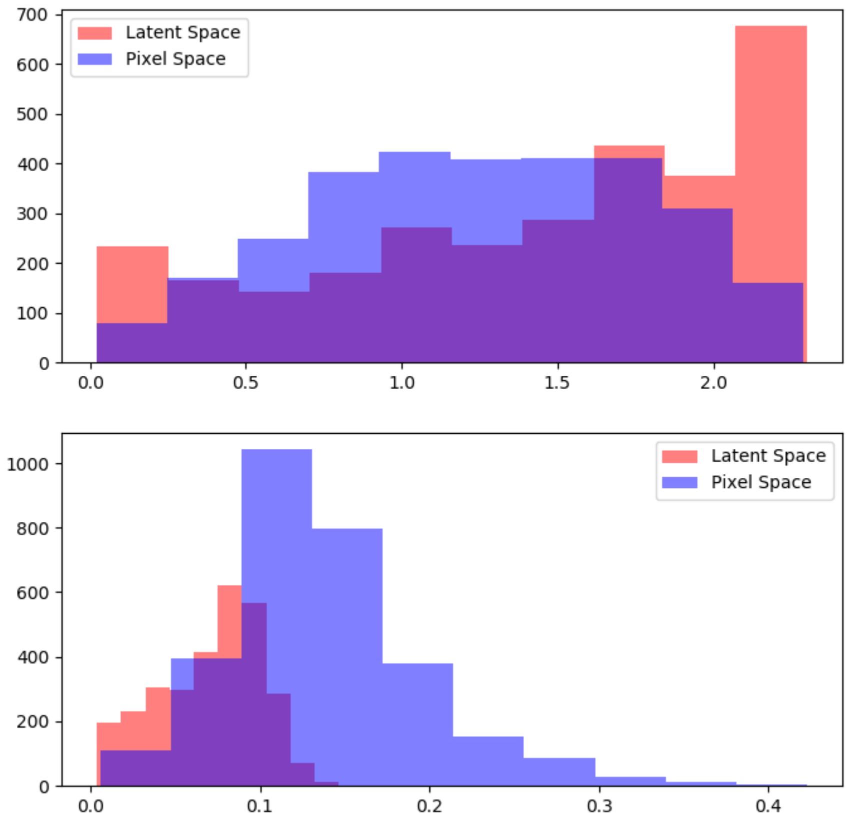


Figure 2.3: The entropy (top) and mutual information (bottom) of the interpolation halfway between 3000 random points of different classes in the MNIST test set, showing that the two modes of interpolation have very different statistical properties with respect to the model uncertainty, as shown for a single example in Figure 2.2.

they do not capture the full posterior, instead capturing local behaviour near one of its modes, since we would expect the uncertainty to be high for a neural network everywhere where it is not constrained by the training data due to the high capacity of the model.

This may offer an explanation as to why MC dropout nets are still vulnerable to adversarial attack; despite their treatment of uncertainty, there are still large regions where they are mistakenly overconfident due to the approximations used, which adversarial attack algorithms can exploit. It may be possible to deliberately find and exploit these 'holes' to create adversarial examples. This intuition suggests a simple fix; since a single dropout model averages over a single mode of the posterior, we can capture the posterior using an ensemble of dropout models using different initialisation, assuming that these will converge to different local modes. We find

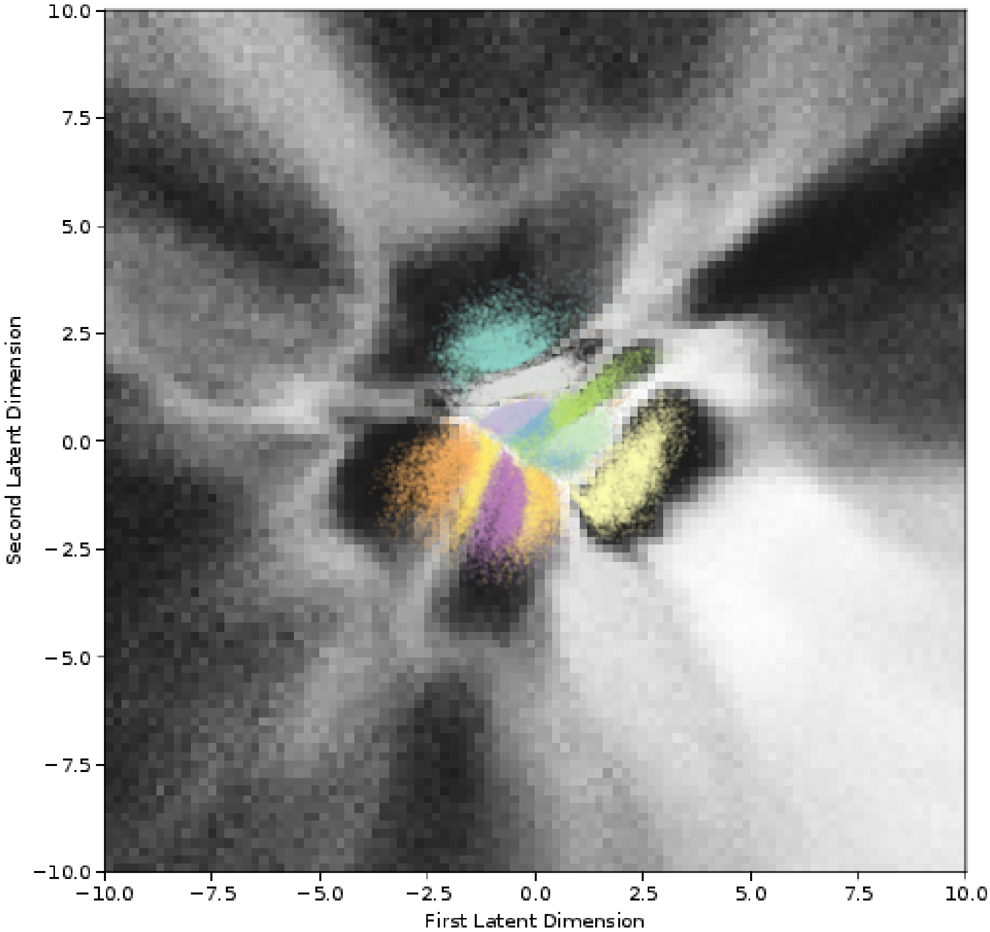


Figure 2.4: The predictive entropy of the same network as in figure 2.1. Note the differences with the MI, which is low everywhere close to the data in the centre of the plot, but the entropy is high between the classes here. These points correspond to images which resemble digits, but which are inherently ambiguous. Note however that there are large regions of latent space where the predictive entropy is high and the MI low, despite being far from any training data.

that even a small ensemble can qualitatively improve this behaviour (figure 2.6).

It should be noted, though, that there is no guarantee that an ensemble of dropout models is a better approximation to the true posterior. It will approximate it well only if the posterior is concentrated in many local modes, all of roughly equal likelihood (since all the models in the ensemble are weighted equally), and a randomly initialised variational dropout net trained with some variant of gradient descent will converge to all of these modes with roughly equal probability[3]. Investigating

[3]This description does coincide with common beliefs about neural network loss surfaces, for which there is some justification in the literature; see, for example, Choromanska et al., 2015

Figure 2.5: A typical garbage class example from the 'holes' in latent space. This is classified as a 2 with high confidence.

possible theoretical justification for this ensembling procedure for variational models is a possible direction for future research.

2.4.3 Evaluation on ASIRRA dataset

It has been observed by (Carlini and Wagner, 2017a) that many proposed defences against adversarial examples fail to generalise from MNIST. Therefore, we also evaluate the various uncertainty measures on a more realistic dataset; the ASSIRA cats and dogs dataset (see Figure 2.7 for example images). The task is to distinguish pictures of cats and dogs, and was originally intended for use in CAPTCHAs[4]. While this is not a state of the art problem, these are realistic, high resolution images. We finetune a ResNet model (He et al., 2015a), pre-trained on Imagenet, replacing the final layer with a dropout layer followed by a new fully connected layer. We use 20 forward passes for the Monte Carlo dropout estimates. We use dropout only on the layers we retrain, treating the pre-trained convolutions as deterministic.

[4]Completely Automated Turing test for telling Computers and Humans Apart, an interactive widget on a website used to prove that the user is a human

We compare the receiver operating characteristic (ROC) of the predictive entropy of the deterministic network, the predictive entropy of the dropout network (equation 2.6), and the MI of the dropout network (the MI is always zero if the model is deterministic; this corresponds to the approximating distribution q being a delta function). Note that we compare with the *same set of weights* (trained with dropout) – the only difference is whether we use dropout at test time. For each measure of uncertainty we generate the ROC plot by thresholding the uncertainty at different values, using the threshold to decide whether an input is adversarial or not.

The receiver operating characteristic is evaluated on a synthetic dataset consisting of images drawn at random from the test set and images from the test set

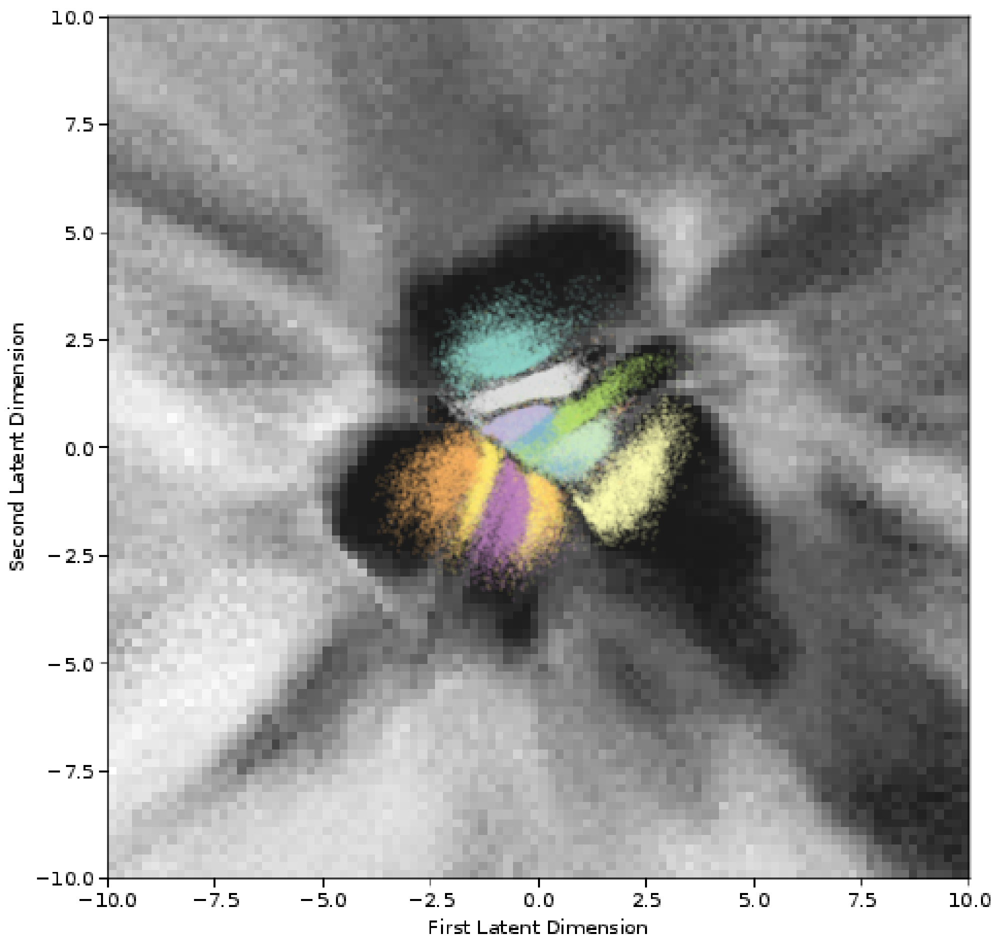


Figure 2.6: The MI calculated using an ensemble of dropout models, treating all of their predictions as Monte Carlo samples from the posterior. This mitigates some of the spuriously confident regions in latent space

corrupted by Gaussian noise, which comprise the negative examples, as well as adversarial examples generated with the Basic Iterative Method (Kurakin et al., 2016), Fast Gradient method (Goodfellow et al., 2014), and Momentum Iterative Method (Dong et al., 2018). We test with the final attack because it is notably strong, winning the recent NIPS adversarial attack competition, and is simpler to adapt to stochastic models than the other strong attacks in the literature, such

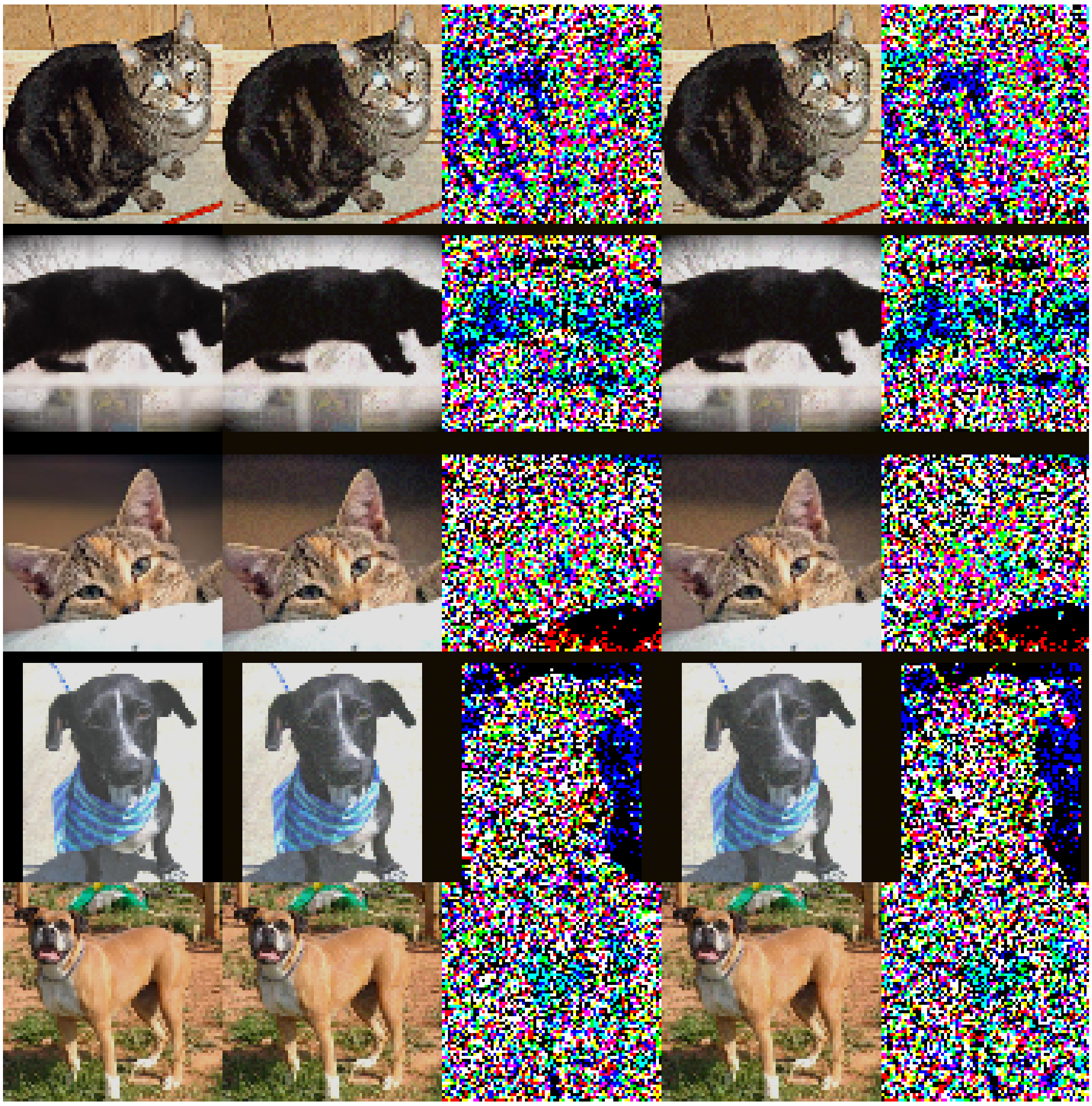


Figure 2.7: Example adversarial images generated by the Momentum iterative method at $\epsilon = 10$, with original images on the left, adversarial images on the deterministic model in the second column, and those for the MC dropout model in the fourth column. The difference between the adversarial image and the original is shown on the right of each image.

	entropy	MI	entropy (S)	MI (S)	Success Rate
BIM $\epsilon = 5$					
Deterministic	0.322	N.A	0.293	N.A	0.757
MC	0.0712	**0.728**	0.0617	**0.733**	0.900
FGM $\epsilon = 5$					
Deterministic Model	0.439	N.A	**0.490**	N.A	0.517
MC Model	0.426	**0.557**	0.465	**0.497**	0.563
MIM $\epsilon = 5$					
Deterministic Model	0.347	N.A	0.319	N.A	0.743
MC Model	0.0476	**0.657**	0.0410	**0.669**	0.917
BIM $\epsilon = 10$					
Deterministic Model	0.302	N.A	0.285	N.A	0.753
MC Model	0.0686	**0.708**	0.0719	**0.723**	0.917
FGM $\epsilon = 10$					
Deterministic Model	0.502	N.A	**0.550**	N.A	0.487
MC Model	0.480	**0.529**	0.514	0.491	0.547
MIM $\epsilon = 10$					
Deterministic Model	0.350	N.A	0.319	N.A	0.763
MC Model	0.0527	**0.661**	0.0442	**0.665**	0.907

Table 2.1: The AUC for the adversarial discrimination task described in the experiments section. Fields marked with (S) denote this quantity evaluated on a version of the dataset with unsuccessful adversarial examples (that do not change the label) removed. The success rate of each attack in changing the label is given as a measure of each attacks effectiveness on this dataset.

as that of Carlini and Wagner (Carlini and Wagner, 2017b).

We find that only the mutual information gets a useful AUC on adversarial examples. In fact, most other measures of uncertainty seem to be worse than random guessing; this suggests that this dataset has a lot of examples the model considers to be ambiguous (high aleatoric uncertainty), which mean that the entropy has a high false positive rate. The fact the AUC of the entropy is low suggests that the model is actually *more* confident about adversarial examples than natural ones under this measure.

An interesting quirk of this particular model is that the accuracy of using Monte Carlo estimation is lower than the point estimates, even though the uncertainty estimates are sensible. Possibly this is because the dropout probability is quite high; only a subset of the features in the later layers of a convnet are relevant to cat and dog discrimination, so this may be a relic of our transfer learning procedure; dropout does not normally have an adverse effect on the accuracy of

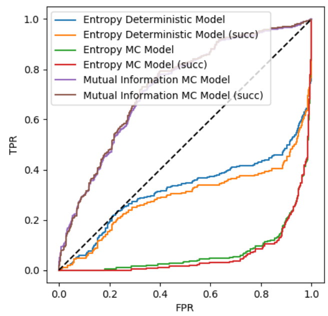


Figure 2.8: BIM with $\epsilon = 5$

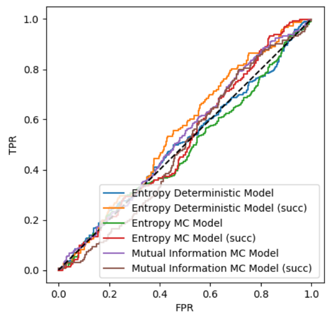


Figure 2.9: FGM with $\epsilon = 5$

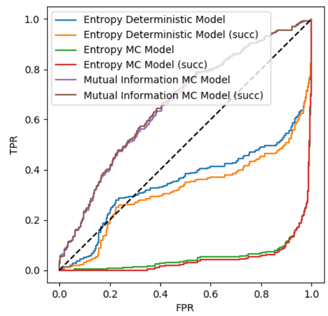


Figure 2.10: MIM with $\epsilon = 5$

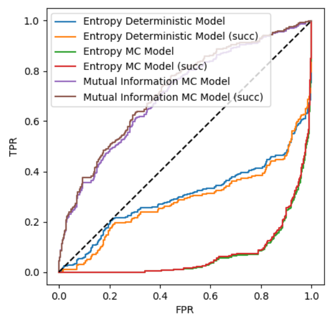


Figure 2.11: BIM with $\epsilon = 10$

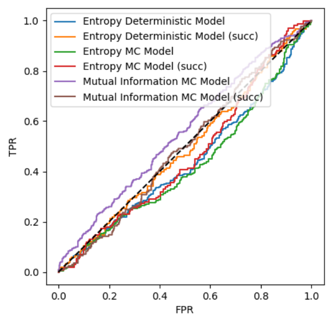


Figure 2.12: FGM with $\epsilon = 10$

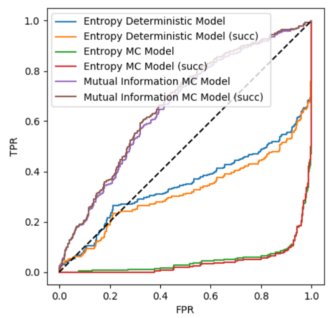


Figure 2.13: MIM with $\epsilon = 10$

Figure 2.14: ROC plots for adversarial example detection with different measures of uncertainty and different attacks. From left to right: basic iterative method (BIM), fast gradient method (FGM), and momentum iterative method (MIM). Top row uses ϵ of 5, bottom row uses ϵ of 10. All use infinity norm. (succ) denotes the quantity evaluated only for successful adversarial examples. We suspect that the low FGM attack success rate is related to the difficulty we observe in identifying these using model uncertainty, however further investigation is required.

fully trained models (Gal, 2016).

2.5 Discussion & Conclusion

This section examined various measures of uncertainty for detecting adversarial examples, and provided both theoretical and experimental evidence that measuring the epistemic uncertainty with the mutual information is the most appropriate and effective for this task.

We used dropout networks to investigate this. We showed that they provide an improvement to standard deterministic models, and that using the appropriate measure of uncertainty improves the performance of this scheme as an adversarial defence.

Taking a broader view, however, the results in this chapter (in agreement with previous literature on the subject) show that dropout networks are *more difficult* to attack than their deterministic counterparts, but attacks against them can still succeed while remaining imperceptible to the human eye, at least in the white-box setting we investigated. In addition, the uncertainty out of distribution clearly contains large regions of erroneous confidence, as visualised in Figure 2.1. As such it is difficult to claim that approximate Bayesian approaches to standard neural networks resolve the problem of adversarial vulnerability.

It is worth noting, however, that these techniques for quantifying uncertainty can be derived without any explicit reference to the adversarial setting, and no assumptions are made about the distribution of adversarial examples. By improving model robustness and dealing with uncertainty more rigorously, models become harder to fool as a side effect; model robustness and good uncertainty estimates are not independent, as discussed in Section 2.3. We think the fact that dropout models can still be defeated by adversarial attack is at least partly because dropout is a fairly crude approximation that underestimates the uncertainty significantly, as we have demonstrated here.

In principle, this problem could be resolved by better approximations. However, another interpretation of the results described in this chapter is that *local* variational approximations, like dropout, do not fundamentally change the extrapolation behaviour of neural networks, as clearly shown in our simple visualisation. Even ensembling dropout models, while significantly better, is not a total solution. While in principle marginalisation over neural network weights would yield a powerful and general model with the OOD behaviour we want, in practice this is not possible to achieve with any known methods, and solutions such as HMC or ensembling in practice cannot use more than a few samples for reasonably sized models. If we want models which reliably revert to some controlled prior outside the data distribution, then, it seems likely we will need to explore non-standard architectures.

3

Adversarial Examples in Idealised Bayesian Neural Networks

Contents

Authorship Statement

The experiments and some of the literature review in this chapter are based on those done by the author for the draft paper "Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with Bayesian neural networks", Gal and Smith, 2018, *arXiv preprint arXiv:1806.00667*. However, the emphasis of the text has been changed significantly, the theoretical argument (which was not my contribution) largely removed, and the discussion on possible explanations for adversarial examples updated to reflect literature published since that draft was written, so this chapter is substantially my own work

3.1 Possible Explanations for Adversarial Examples

In the previous chapter, we discussed how standard Bayesian methods behave under adversarial perturbation. In this chapter, we discuss in more detail possible explanations in the literature for *why* deep models appear to be so prone to adversarial attack. We then discuss some experiments with idealised Bayesian models, exploring whether ideal inference algorithms would resolve this problem, and discuss how practical these would be. Finally, we discuss why these results motivated the non-standard deep models we investigate in the remainder of the thesis.

When first introduced in Szegedy et al. (2013), adversarial examples were initially hypothesised to be similar to the rational numbers, a dense set within the set of all images, due to the nonlinearity of deep networks. Essentially, these were considered to be singular points which occur due to the complexity of the functions neural networks learn. Szegedy et al. (2013)'s gradient based method for generating adversarial images performed a *targeted* attack, where a small perturbation added to an image causes the network to classify differently to the original image class. Follow-up research by Goodfellow et al. (2014) introduced *non-targeted* attacks, where a given input image is perturbed to an arbitrary incorrect class by following the gradient away from the image label. This technique also gave rise to a new type of adversarial examples, "garbage" images, which look nothing like the original training examples yet classify with high output probability. Goodfellow et al. (2014) showed that the deep neural networks' non-linearity is not the cause of vulnerability to adversarial examples, by demonstrating the existence of adversarial examples in linear models as well. They hypothesised that NNs are very linear by design and that in high-dimension spaces this is sufficient to cause adversarial examples, in contrast to the idea that adversarial examples only happen because neural networks learn *complicated* functions. Later work studied the linearity hypothesis further by constructing linear classifiers which do not suffer from the phenomenon (Tanay and Griffin, 2016). The idea that adversarial examples are somehow singular points can also be refuted by work showing that adversarial examples can be created on physical media and still produce adversarial images when photographed (Kurakin et al., 2016), demonstrating that the adversarial perturbations are robust to the distortions induced by this process. While they are not always considered adversarial, the visualisations in the previous chapter showing large continuous regions in the latent space of a generative model which are classified with high confidence despite decoding to 'garbage' that does not resemble natural images are also indicative of the fact that this is not a phenomenon confined to isolated points.

Another question is whether adversarial images are 'unnatural', in the sense of having low probability under the data distribution. This is related to the idea that natural images form a low-dimensional manifold in the input space: every natural image could be close, in Euclidean distance, to some point which is out of the data distribution, and on which models trained on standard data would not be expected to generalise well. Nguyen et al. (2015) developed techniques which do not rely on gradients but rather use genetic algorithms to generate "garbage" adversarial examples. Nguyen et al. (2015) further hypothesised that such adversarial examples have low probability under the data distribution, and that joint density models $p(\mathbf{x}, y)$ will be more 'robust' because the low marginal probability $p(\mathbf{x})$ would be indicative of an example being adversarial.

Li et al. (2019) recently extended these ideas to non-garbage adversarial examples as well, and lent support to the hypothesis by showing on MNIST that a deep naive Bayes classifier (a generative model) is able to detect targeted adversarial examples by thresholding low input density. While it is difficult to empirically verify as there is no objective way of determining the probability of a particular image under the natural data distribution, intuitively this theory makes a certain amount of sense, as it potentially explains why adversarial examples seem so easy to find for models which appear to perform extremely well on natural images.

It is worth noting that more recently, generative models which allow both sampling from more complicated distributions and evaluating the likelihood of images have been designed, and there are in fact some issues with using the density to evaluate points as in or out of distribution which are described in Nalisnick et al.

(2019a). In particular, they demonstrate that the likelihood of datasets is intrinsically linked to their variance, which can confound comparisons of datasets based on the likelihood of a learned generative model even if the model is a reasonable one for the dataset in question. While this does not necessarily mean that considering the 'true' distribution of the data is not a useful thought experiment, it does mean that a direct approach to trying to model the density of the true distribution is not the obvious solution that it might initially appear to be.

Setting aside the difficulties of the model density, it is not in fact clear that adversarial examples can only exist 'off' the data distribution. Gilmer et al. (2018) construct a simple dataset composed of a uniform distribution over two concentric spheres in high dimensions, with a deterministic feed-forward NN trained on 50M random samples from the two spheres. They use constrained optimisation to search for adversarial on one of the two concentric spheres, i.e. in a region of high density, and demonstrate that this attack successfully finds adversarial examples with a model trained on the spheres dataset. Whether this really qualifies as 'adversarial' (as opposed to simply missclassified) points is an interesting question, but it again demonstrates that the link between likelihood under the data distribution and the existence of missclassified points is not straightforward, as it falsifies the hypothesis that adversarial examples must exist in low density regions of the input space.

A parallel line to the above research has tried to construct bounds on the minimum magnitude of the perturbation required for an image to become adversarial. Fawzi et al. (2018) for example quantify "robustness" using an introduced metric of expected perturbation magnitude and derive an upper bound on a model's robustness. Fawzi et al. (2018)'s derivation relies on some strong assumptions, for example assuming that it is feasible to compute the distance between an input $\mathbf{x}$ and the set $\{\mathbf{x} : f(\mathbf{x}) > 0\}$ for some classifier $f(\mathbf{x})$. Papernot et al. (2016) further give a definition of a robust model, extending the definitions of Fawzi et al. (2018) to targeted attacks, and propose a model to satisfy this definition. In more recent work, Peck et al. (2017) extend on both these ideas, and propose a lower bound on the robustness to perturbations necessary to change the classification of a neural network. Peck et al. (2017) also make strong assumptions in their premise, assuming the existence of an oracle $f^*(\mathbf{x})$ able to assign a "correct" label for each input $\mathbf{x} \in \mathbb{R}^D$. This assumption is rather problematic since it implies that any input has a "correct" class, including completely blank images which have no objects in them. Lastly, Hein and Andriushchenko (2017), working in parallel to (Peck et al., 2017), use alternative assumptions and instead offer a bound relying on local Lipschitz continuity.

More recent theoretical work has suggested an intriguing alternative; that perhaps adversarial examples are not due to the model itself, but arise in part due to the geometrical properties of high dimensional spaces. Shamir et al. (2019) show that any piecewise linear function, such as a ReLU convolution network, which partitions an input space $\mathbb{R}^n$ into cells assigned to a small number of m classes, must have adversarial examples with a small Hamming distance. They describe a constructive algorithm for a trajectory going from a point x cell i to any other cell j while only changing around m of the entries of x. While this argument does not generalise to adversarial examples in other norms, in is important to bear in mind; the geometry of mapping a high dimensional input space to a small number of classes is not particularly intuitive.

Even more recently, Bubeck and Sellke (2021) argue that there are geometrical constraints on the complexity of model required to learn a *robust* model in high dimensions. They establish that, while to fit a function which perfectly interpolates n datapoints in $\mathbb{R}^d$ requires only n parameters, interpolating a *Lipschitz smooth* function between n datapoints requires $\mathcal{O}(nd)$ parameters.[1] This could mean that adversarial examples occur in models which are *under* rather than over parameterised! Interestingly, this theory roughly predicts the order of magnitude of parameter counts at which deep models begin to extrapolate robustly on small datasets like MNIST. A rough calculation based on this bound would suggest that

[1]This offers an interesting possible explanation of the spheres dataset. Given a mesh of n points equally spaced on the surface of the sphere, it ought to be possible to ensure that there are no misclassified points by ensuring a model has a given Lipschitz constant. This theory would predict that a sufficiently overparameterised model could do this, which would be an interesting experiment.

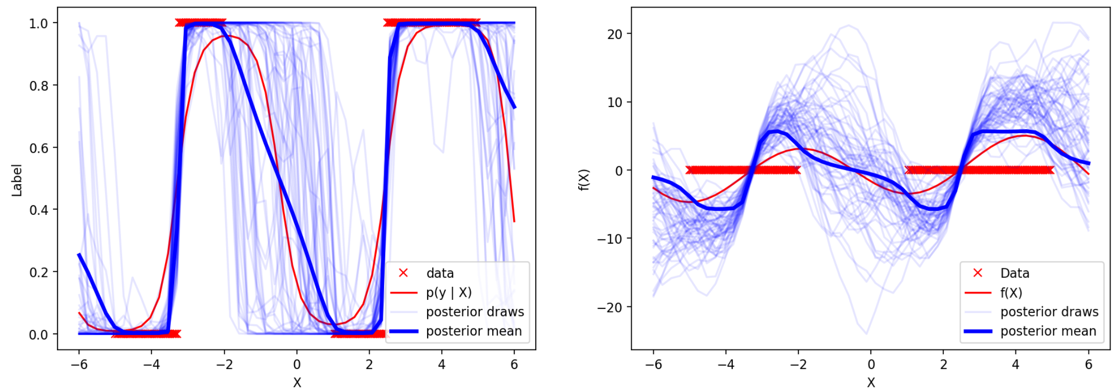


Figure 3.1: Idealised Bayesian inference. The ensemble of functions agrees on the dataset, but produces arbitrary values outside it; in the ideal Bayesian case, we would expect high uncertainty anywhere the data density is low

on larger datasets, such as Imagenet, even modern models (at the time of writing) possibly do not have sufficient parameters to learn a robust function, in the sense of being Lipschitz bounded; a function with a *high* Lipschitz constant clearly has something approximating adversarial examples, as a small change to the input can have a large effect on the output of the function.

3.2 Idealised Models

We have discussed a variety of hypothesis for why adversarial examples occur. Some of these are related to intrinsic properties of standard neural networks. For example, the linearity hypothesis of Goodfellow et al. (2014) is related to the intrinsic way in which a standard neural network generalises in a linear fashion, and the explanation in Shamir et al. (2019) is related to the piecewise linearity of the models.

It is worth considering whether idealised Bayesian inference would resolve these problems.

Figure 3.1 shows idealised Bayesian inference for a neural network in a 1D setting. This gives an intuition behind an common assumption of ideal Bayesian inference, namely that our uncertainty should increase uniformly anywhere away from the training data.

If our prior is sufficiently flexible, for instance, if we consider all functions with a given Lipschitz constant L as a-priori equally likely, then it is obvious that simply by continuity, as we move a distance ϵ away from our training datapoints, then we expect our uncertainty to increase. In this case, the model uncertainty would be a reasonable proxy for the density under the training distribution, or the distance to the closest training point.[2] Such a model is not too difficult, in theory, to construct, though it may not be efficient to use; for instance, a Gaussian process with a radial basis function kernel behaves in this way. Alternatively, this can be added in a post-hoc manner by fitting a simple density function to the data points, and rejecting points which fall outside it; this approach was proposed, for example, in Meinke and Hein (2019).

There are two important considerations here. The first is that such a model may not generalise well. A motivating example here is the sphere dataset of Gilmer et al. (2018); a model which reports high uncertainty above a given distance from all training points will have no adversarial examples on the sphere, but it may also not perform well unless the training data covers the sphere.

From this, we see that the model incorporating the relevant *invariances* of the training set is also an important consideration for an idealised model. Incorporating the invariances of the training set could potentially lead to robustness without any negative effects on generalisation. This idea is not particularly explored further in this chapter, but was an important motivation for the work on capsule networks described in chapters 4 and 5.

An important caveat to this, of course, is the fact that standard neural networks are *not* idealised models. It is worth investigating empirically whether the assumption that neural networks are flexible enough function classes that averaging over them will increase uncertainty everywhere far from the data is a reasonable one; it is very possible that the prior implied by the prior over weights and the network architecture is far more structured than this. We experimentally examine this in the following section.

[2] This is explicitly true in non-parametric models. Neural networks, of course, are parametric models and so in the limit of data their uncertainty should saturate. It is commonly assumed with deep learning that we are working in a regime where the number of model parameters is larger that the number of datapoints so this does not apply, but it is worth bearing in mind.

3.3 Empirical Evidence

In this section we give empirical evidence supporting the arguments above. We demonstrate the ideas using near-perfect epistemic uncertainty obtained from HMC (considered 'gold standard' for inference with BNNs (Neal, 1995)), and with image data for which we know the ground-truth image-space density.[3] We show that, in this model, image density diminishes as images become adversarial, that uncertainty correlates with image density, and that state-of-the-art adversarial attack methods fail with HMC. We then test how these ideas transfer to non-idealised data and models, demonstrating failures of dropout uncertainty.

3.3.1 Idealised case

As mentioned, one popular hypothesis for adversarial examples is that they are 'unnatural', outside the data distribution. Though as shown by Gilmer et al. (2018), this is not universally the case, it is interesting to consider whether it holds in more realistic situations. However, it is impossible, without making assumptions about the generative process, to quantify the probability density of a particular image, making the claim that adversarial examples have low probability difficult to quantify. To address this, we derive a new, synthetic, image dataset from MNIST (LeCun et al., 1998a), for which we can calculate the ground truth density in the image space, and are therefore able to determine how far away it is from the data distribution.

Our dataset, Manifold MNIST (MMNIST) was constructed as follows. We first trained a variational auto-encoder (VAE) on MNIST with a 2D latent space. We chose three image classes[4] (0, 1, and 4), discarding the latents of all other classes, and put a small 'Gaussian bump' on each latent point from our 3 classes. Summing the bumps for each latent class we get an analytical density corresponding

[3]Using a synthetic dataset means that this avoids some of the issues identified in the literature with learning generative models.

[4]We wanted to use a model with a low dimensional latent space in order to make visualisation and marginalisation of the latent variable easier, but this obviously limits the ability of the model to accurately model the classes. As this is a synthetic dataset, how closely it resembles actual MNIST is not that important, but we choose classes which are reconstructed with reasonable accuracy and have little overlap in latent space to reduce class ambiguity in our synthetic dataset

to this class. We then discarded the MNIST latents, and defined the mixture of the 3 analytical densities in latent space as our ground truth image density (each mixture component identified with its corresponding ground truth class). We refer this to dataset as 'Manifold MNIST' because it is - by construction - restricted to a complex manifold in image space, defined by the push-forward of the latent space density through the learned decoder. Generating 5,000 samples from this mixture and decoding each sample using our fixed VAE decoder, we obtained our training set for which each image has a ground truth density (Figure 3.2). A variational autoencoder does not permit analytical density calculation, but since the latent space is low-dimensional by design we can calculate likelihoods in data space by marginalising the latent variable.

First we show that the density decreases on average for image $\mathbf{x} \sim \texttt{MMNIST}$ as we make $\mathbf{x}$ adversarial (adding perturbations) using a LeNet classifier (LeCun et al., 1998b). Multiple images were sampled from our synthetic dataset, with the probability of an image in the input space plotted as it becomes adversarial for both targeted and non-targeted FGM (Goodfellow et al., 2014) attacks (Figure 3.3). We find that, at least for the initial steps of the attack, the deformations remain imperceptible, even as the likelihood of the images under the generative model is significantly reduced.

Next, we show that 'gold standard' epistemic uncertainty, as calculated using HMC to integrate out the network weights, correlates to density under the image manifold, in our synthetic dataset. We use $\mathcal{D}_{\text{grid}}$ given by a grid of equally spaced points over the 2D latent space (Figure 3.4) to sample points with a high and low likelihood under the data generating distribution, in order to verify if the MI is a reasonable proxy for the density. Figure 3.5, visualises this uncertainty in the VAE latent space; in white is uncertainty, calculated by decoding each latent point into image space, and evaluating the MI between the decoded image and the model parameters; A lighter background corresponds to higher uncertainty), demonstrating that the HMC uncertainty is a good proxy for the support of the

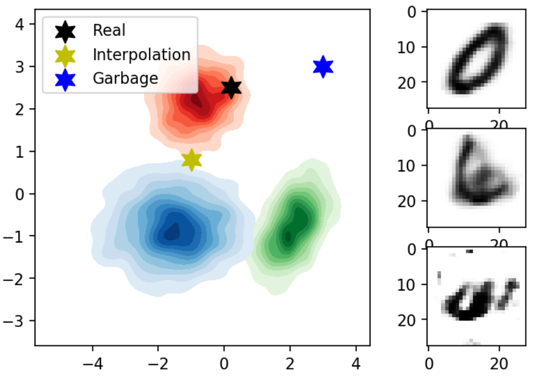
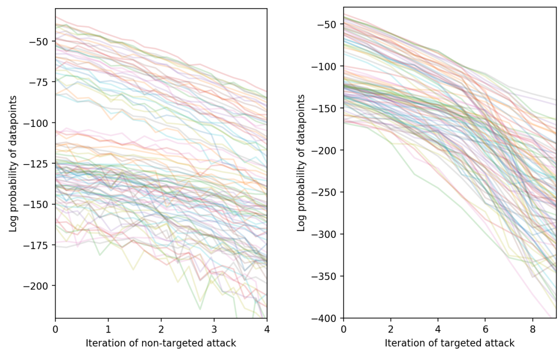


Figure 3.2: Manifold MNIST ground-truth density in 2D latent space with decoded image-space realisations (a real-looking digit (top), interpolation (middle), and garbage (bottom)).



(a) Non-targeted **(b)** Targeted

Figure 3.3: Ground-truth density v.s. step for FGM attacks on the decoded images with a deterministic classification NN. **Note the decreasing density as the images become adversarial**.

dataset in latent space. In Figure 3.6 we show that uncertainty correlates to density on the images from $\mathcal{D}_{\text{grid}}$.

Finally, we show that methods for finding adversarial examples fail for HMC. In this experiment we sample a new realisation from the HMC predictive distribution with every gradient calculation, in effect approximating the infinite ensemble defined by an idealised BNN. We used a non-targeted attack (MIM, first place in the NIPS 2017 competition for adversarial attacks (Dong et al., 2018)), which was shown to fool finite deterministic ensembles and be robust to gradient noise[5]. Table 3.1 shows success rate in changing test image labels for HMC and a deterministic NN, for maximum allowed input perturbation of sizes $\epsilon \in \{0.1, 0.2\}$, v.s. a control experiment of simply adding noise of magnitude ϵ. Note HMC BNN success rate for the attack is similar to that of the noise, v.s. Deterministic where random noise does not change prediction much, but a structured perturbation fools the model *very* quickly. Note further that HMC BNN's entropy increases quickly, showing that the model has many different possible output values for the perturbed images.

[5]This is important because one way to make a model more difficult to attack is to simply make the gradient estimates noisy, which makes it harder for optimisation algorithms based on gradients to find adversarial examples, even if they exist. Using HMC in this way clearly introduces noise to the gradient estimates, so using a method which is robust to noise should mitigate this, though we can't rule out that this is why HMC is more difficult to attack.

	HMC BNN				Deterministic NN			
	Adv. succ.	Noise succ.	Adv. $\mathcal{H}$	Noise $\mathcal{H}$	Adv. succ.	Noise succ.	Adv. $\mathcal{H}$	Noise $\mathcal{H}$
$\epsilon = 0.1$	0.14 ± 0.03	0.10 ± 0.01	0.47 ± 0.00	0.33 ± 0.00	0.52 ± 0.0	0.03 ± 0.001	0.45 ± 0	0.06 ± 0
$\epsilon = 0.2$	0.32 ± 0.02	0.23 ± 0.01	0.59 ± 0.02	0.53 ± 0.01	0.97 ± 0.0	0.03 ± 0.002	0.03 ± 0	0.08 ± 0

Table 3.1: MIM untargeted attack with two max. perturbation values ϵ, applied to HMC BNN and Deterministic NN, showing success rate (*lower is better*) in changing true label for attack, changing true label by adding noise of same magnitude (control), and showing average entropy $\mathcal{H}$ for perturbed images. Shown mean $\pm$ std with 5 experiment repetitions.

3.3.2 Non-idealised case

Here we compare real-world inference (specifically, dropout) to near-perfect inference (HMC) on real noisy data (MNIST). We use the same encoder as in the previous section to visualise the model's epistemic uncertainty in 2D (Figure 3.7). Note the dropout uncertainty 'holes' compared to HMC. We plot the dropout MI v.s. HMC MI for the grid of points $\mathcal{D}_{\text{grid}}$ as before in Figure 3.8.

3.3.3 Real-world cats vs dogs classification

We extend the results above and show that an ensemble of dropout models is more robust than a single dropout model using a VGG13 (Simonyan and Zisserman, 2014) variant on the ASIRRA (Elson et al., 2007) cats and dogs classification dataset, as used in the previous chapter. We retrained a VGG13 variant ((Simonyan and Zisserman, 2014), with a reduced number of FC units) on the ASIRRA (Elson et al., 2007) cats and dogs classification dataset, with Concrete dropout (Gal et al., 2017) layers added before every convolution. We compared the robustness of a single Concrete dropout model to that of an ensemble following the experiment setup of the previous chapter. Here we used the FGM attack with $\epsilon = 0.2$ and infinity norm. Example adversarial images are shown in Figure 3.12. Table 3.2 shows the AUC of different MI thresholds for declaring an input adversarial for all images, as well as for successfully perturbed images only (S). Full ROC plots are given in Figure 3.13. We note that the more powerful attacks succeed in fooling this VGG13 model, whereas *dropout Resnet-50* based models seem to be more robust. We leave the study of model architecture effect on uncertainty and robustness for future research.

Model	AUC	AUC (S)
Concrete Dropout	0.63	0.61
Concrete Dropout Ensemble	0.77	0.74

Table 3.2: AUC for MIM attack on VGG13 models (*higher is better*), trained on real-world cats v.s. dogs classification, for both a single Concrete dropout model, as well as for an ensemble of 5 Concrete dropout models

3.4 Discussion

We have discussed several hypotheses in the literature for the lack of robustness of neural networks under adversarial perturbation. We think there are two of these which are the most plausible;

a) Adversarial examples are due to over-extrapolation into regions of low data density. While some evidence shows this is not the only cause, our experiments with manifold MNIST show that it may still be a contributing factor in practical situations.

b) Adversarial examples are an unintuitive feature of high dimensional geometry, which are likely to be present in any model which maps a high dimensional space into a small number of classes

These explanations are not necessarily mutually contradictory, but they do have importantly different implications. If adversarial examples are mainly due to over-extrapolation, then improving the inference methods, or modifying the architecture of the network, or changing the model assumptions may largely resolve adversarial examples, though this has not been particularly successful so far. On the other hand, if adversarial examples are at least largely a geometrical phenomenon, it may be that adversarial examples are fundamentally not avoidable for models performing certain tasks, like classifying images into a small number of classes.[6]

[6] One might object that more sophisticated models may, in the future, move away from this paradigm and be able to give more detailed descriptions of images, such as textual descriptions. However, ultimately, if a model is used for downstream decisions, the number of effective classes may still be fairly small, corresponding to the number of decisions. For example, a model which is used to moderate images on social media may ultimately only have a few possible actions (maybe allow, ban, flag) meaning that presumably adversarial attacks on these decision classes would be possible.

The lack of general progress at avoiding adversarial examples robustly in a practical way since they were introduced may suggest that b) plays at least a significant role.

The possibility outlined in Bubeck and Sellke (2021), that adversarial examples may be due to *under* parameterisation, is a very interesting alternative, but this is very recent work and further research is likely required on this.

Given these open questions about adversarial examples, we do not focus on this particular application in the remainder of the thesis, focussing instead of non-adversarial out of distribution performance. However, we still hope to improve on the out of distribution performance of standard architectures at a comparable computational cost.

While our experiments above showed that *idealised* inference, in the form of HMC, significantly improves the robustness of model uncertainty on out of distribution and even adversarial images, HMC is not an especially practical method of mitigation due to its high computational cost.

Much work has gone into improving approximate inference in Bayesian neural networks, and the experiments in this chapter suggest that they could improve things if they were powerful enough, like HMC. But while we have only conducted experiments with dropout, most tractable approximations resemble dropout in the sense that they are *local*, exploring only a single mode of the true posterior, and our results with this kind of model are less promising. We did find that ensembling these local approximations can achieve significant improvements. Ensembles of deterministic models have also proved to be an effective technique in practice (Lakshminarayanan et al., 2017), though we didn't investigate them in this chapter. While ensembling is a simple and practical technique, it still has drawbacks, most significantly in the added computational cost of training and inference.

The rest of the thesis explores an alternative approach to simply trying to increase the expressivity of approximate inference. The following chapters of the thesis are based on two key ideas which address the possible causes of adversarial examples: *capsule networks* represent an attempt to enforce the natural invariances of images into a model explicitly, which as discussed above should increase the generalisation of models out of distribution. The bilipschitz models discussed in the last part are an attempt at a more limited goal, namely to ensure prior reversion away from the training data similarly to HMC inference, but at a more practical computational cost.

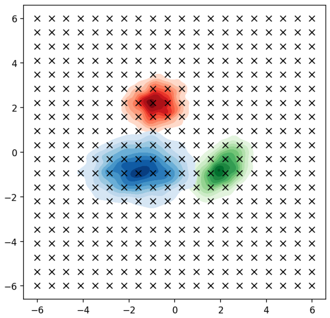


Figure 3.4: Our $\mathcal{D}_{\text{grid}}$ dataset depicted in 2D latent space with crosses overlaid on of Manifold MNIST.

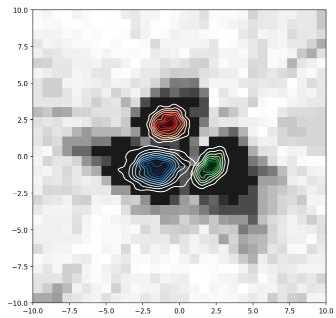


Figure 3.5: Manifold MNIST 2D latent space with HMC MI projected from image space, showing “near perfect” uncertainty.

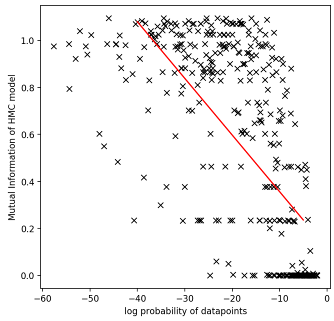
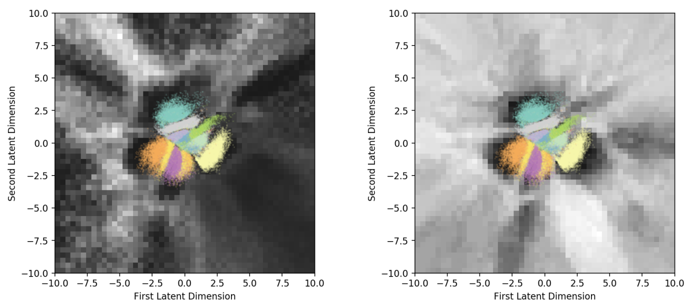


Figure 3.6: HMC MI v.s. log density of $\mathcal{D}_{\text{grid}}$ latent points. **Note the strong correlation between the density and HMC MI.**



(a) Dropout (b) HMC

Figure 3.7: MNIST projected into 2D latent space with projected image-space MI for *dropout* and *HMC* inference. **Note the holes in uncertainty far from the data for dropout**.

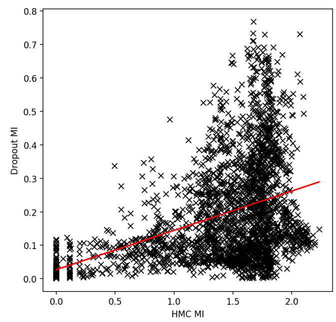


Figure 3.8: Dropout MI v.s. HMC MI (note the correlation, but also that **dropout holes** lead to zero MI when HMC MI is non-zero).

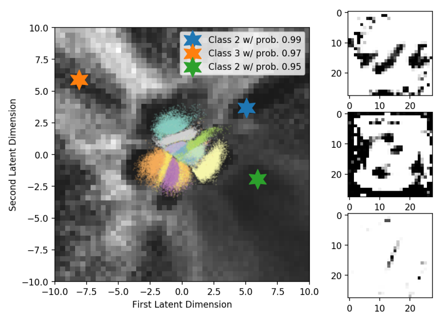


Figure 3.9: Three cherry-picked ‘garbage’ images classifying with high output prob., from ‘holes’ in dropout uncertainty over MNIST, and their locations in latent space.

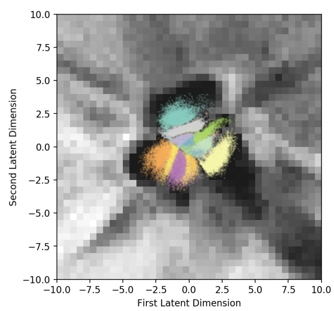


Figure 3.10: 2D latent space with MI of *dropout ensemble* on MNIST, showing **fewer uncertainty ‘holes’** v.s. dropout (3.7a).

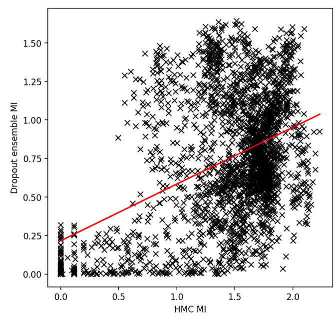


Figure 3.11: Dropout ensemble MI v.s. HMC MI (most of the mass on the right has been **shifted up**, i.e. dropout holes are covered).

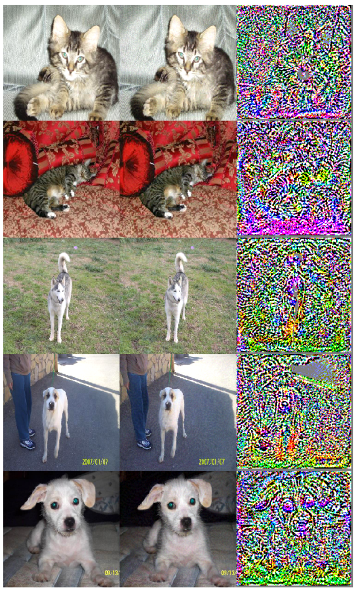


Figure 3.12: Example dataset images, generated adversarial counterparts, and the perturbation.

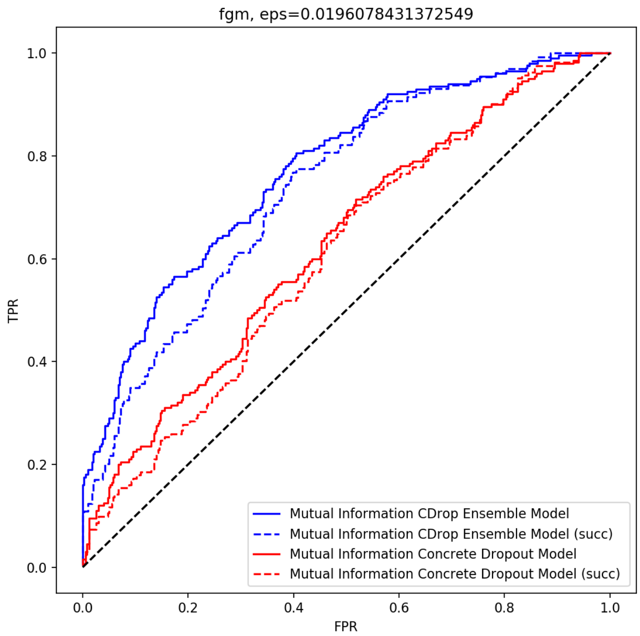


Figure 3.13: ROC plot of dropout and dropout ensemble using MI thresholding to declare 'adversarial', evaluated both on all examples, and on successfully perturbed examples (marked with 'succ').

Part II

Capsule Models

But what are the simple constituent parts of which reality is composed? – What are the simple constituent parts of a chair? – The pieces of wood from which it is assembled? Or the molecules, or the atoms? – "Simple" means: not composite. And here the point is: in what sense 'composite'? It makes no sense at all to speak absolutely of the 'simple parts of a chair'.

Ludwig Wittgenstein, *Philosophical Investigations*

4 Capsule Networks: A Generative Perspective

Contents

4.1 Introduction

Authorship Statement

This chapter is based on the draft paper "Capsule Networks–A Probabilistic Perspective", **Smith**, Schut, Gal, and Wilk, 2020, *arXiv preprint arXiv:2004.03553*, a version of which was published at the ICML Object Oriented Learning workshop in 2020. The majority of this is substantially my own work, but my co-author Lisa Schut provided invaluable help in implementing the experimental comparisons to Kosiorek et al. (2019) using their codebase. Mark van der Wilk had the original idea of an explicitly variational approach to capsule models, and played an important role in shaping my attempts to design such a model.

In the previous section of this thesis, we detailed some undesirable properties of standard network architectures, and outlined some possible theoretical explanations for them. While we were able to resolve these with close to ideal Bayesian inference, as modelled by HMC, approximate Bayesian methods such as dropout shared many of the same drawbacks as standard models, and HMC is computationally prohibitive. This leads us to investigate alternative 'deep' structures to the standard network architectures. This section of the thesis investigates one particular alternative: so called 'capsule' networks.

Capsule networks attempt to incoporate assumed invariances about the structure of scenes into the model, in principle leading to better generalisation. As we argued in Chapter 3, reflecting the natural invariances in a problem is a desirable property for a robust model, and so it is reasonable to hope that a model which reflects these invariances would give better out of distribution behaviour than we observed in standard neural networks. As we shall see in the following two chapters, we don't feel that capsule networks are actually a very promising direction for this going forward, but this was the initial motivation for us to study them in more detail. This section first introduces a novel formulation of capsule models, which attempts to encode the stated assumptions of previous work on capsules in a consistent, generative framework, and then investigates the performance of this model.

Capsule models (Hinton et al., 2018; Sabour et al., 2017) are based on the idea that complex objects can be described as a geometric arrangement of simpler objects. In rigid bodies, the poses of these parts are related to the pose of the whole through an affine transformation. Relative poses can be represented using a matrix R in homogeneous coordinates, which allows the pose A_c of child-objects to be expressed in terms of the parent poses A_m as $A_c = A_p R$. The central idea of capsules is to build a model which begins by inferring poses for simpler objects, then describes the poses of these simpler objects in terms of composite objects, a form of compression since a description like 'there is a bike in a given pose' is more compact that 'there is a handle in pose A, a wheel with pose B, etc.'. The fact that the part-whole relationship is linear should allow models to be able to generalise to recognising the same object from multiple viewpoints, as if we are able to capture a linear relationship this allows strong generalisation due to the simplicity of a linear model.

This intuition behind capsules stems from a generative argument: if we know that an object is present, then the child-objects it is composed of must also be present and have consistent poses that are predicted through the linear relationship above. In this model, a scene is generated by first sampling the high level objects it contains, then conditioned on these the parts that make them up are drawn, and so on. When inferring the presence of an object, this reasoning is inverted. The pose of each child object predicts a pose for the composite object through $A_p = A_c R^{-1}$. An object is likely to be present if the child-objects it is composed of are present *and predict similar poses* for the parent object. If the poses are not consistent, then child-objects are unlikely to have been caused by the presence of a single object.

Sabour et al. (2017) and Hinton et al. (2018) proposed *routing* algorithms for finding consistent poses. The latter finds clusters of consistent poses from child-objects using a procedure inspired by Expectation Maximisation (EM) for Gaussian mixtures. These routing methods are explained informally as performing inference on some *latent variables* in the model, but are not derived from a direct approximation to Bayes' rule in a well specified generative model.

In this chapter, we describe an explicit generative model which formulates these assumptions in a probabilistic framework, showing how a 'routing' algorithm can be derived in this context. We do so with two main aims. Firstly, we aim to express the modelling assumptions of capsules as a probabilistic model with a joint distribution over all latent and observed random variables. This explicit expression through probability distributions and conditional independence allows the model specification to be critiqued and adjusted more easily. Our second aim is to provide a principle by which algorithms for taking advantage of these assumptions can be derived. By phrasing the problem as approximate inference in a graphical model, we have a procedure for deriving routing schemes for modified models, or for taking advantage of the literature on inference in graphical models.

4.2 Capsule Networks

4.2.1 Objects and parts

The assumptions that capsules try to capture about a scene are

1. The scene can be described visually in terms of a set of simple objects, or graphics primitives

2. The positions of the simple objects can be described concisely in terms of complex objects (parents), which are geometrical arrangements of simpler objects (children)

3. Objects can be organised into a hierarchy, with simple objects at the bottom and complex objects higher in the hierarchy

4. Each child has at most one immediate parent—that is, it is a part of at most one higher-level object, although we may be uncertain about which one this is.

Figure 4.1 shows a cartoon of an object (a cup) with two parts, the body and the handle, which obeys the assumptions above. Each object has a pose A, and a relative transformation R, such that the position of the parts of the object relative to the pose of the cup is given by a linear transformation $A_{\text{part}} = A_{\text{whole}}R$. In a full capsule model, there are assumed to be many 'high level' objects such as the cup that these parts may or may not be part of; for instance, the handle part could maybe be explained as part of a saucepan object instead. But the final assumption assumes that in a particular scene, the handle will be part of only one object at a time.

This is interesting from a learning perspective because the part-whole relative pose R is invariant to changes in the pose of the whole object, at least for rigid

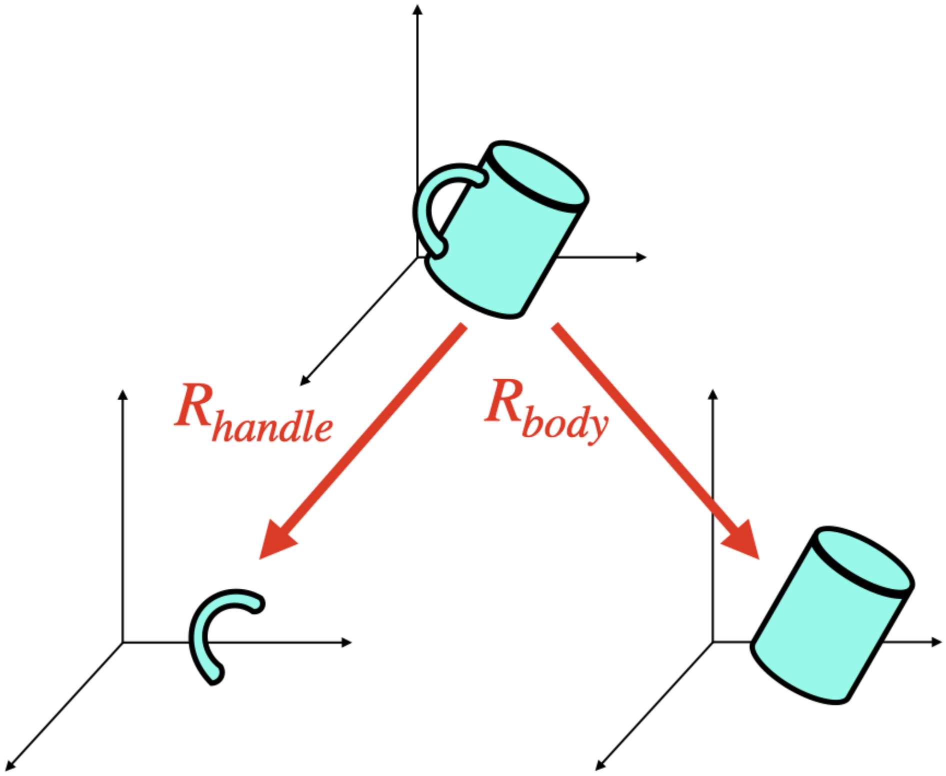


Figure 4.1: A sketch of the relationship between the part-whole orientation and the pose of the object.

objects. We can therefore potentially learn representations which are likewise invariant to changes in pose by exploiting this relationship. Convolutional networks are so effective, in part, because they can exploit the natural invariance to position of a feature detector, so this wider class of invariances is potentially very exciting, since if we can capture it we could potentially learn from smaller amounts of data, as well as being more robust to changes in viewpoint.

4.2.2 A graphical model

Previous work (Hinton et al., 2018; Kosiorek et al., 2019) describe isolated parts of their capsule models using likelihoods. While these likelihoods inspire loss terms in the training objective, the likelihoods do not form a complete generative model over all variables, and the distinction between latent (random) variables and neural architecture choices is not made. From the recent approaches to capsules, Kosiorek et al. (2019) rely most on expressing loss terms as likelihoods but lack a generative expression, for example, over the presences of child objects. We intentionally follow the model structure and intuitions of earlier work but take care to build a probabilistic model with a globally well-defined joint probability distribution over all random variables, allowing us to derive different inference methods for the same graphical model, and consider whether the model itself makes sense independently from the inference technique used.

It is important to note that previous iterations of capsules have used different models, some fully discriminative, and some with a partial or fully unsupervised autoencoding framework. We discuss connections to prior work in more detail in Section 4.3.

We introduce a generative model that captures the assumptions outlined in Section 4.2.1. The generative model is for a single image; different images have their own composition of objects, and therefore have image-specific random variables representing this. In the following, per-image random variables are in black, while global model parameters that are shared across different images are in red. Each object that can be present in an image is represented through its presence t and pose A. Objects are present when $t = 1$, and absent otherwise. A capsule is the presence and pose for an object taken together. To form the hierarchy of objects, we arrange capsules in layers denoting the layer k in the superscript, and the object i in the subscript, i.e. t_i^k, with n_k capsules in a layer. The top layer capsules are independently present at a random pose, encoded as

$$t_i^0 \sim \text{Bernoulli}(p)\,, \qquad 1 \le i \le n_0\,, \tag{4.1}$$

$$A_i^0 \sim \mathcal{N}(0, I)\,, \qquad 1 \le i \le n_0\,. \tag{4.2}$$

Each parent object is composed of a different combination of child objects, so we represent the affinity of a parent i to a child j using the global parameter ρ_{ij}^k, which can be thought of as representing the conditional probability of that child being present and part of that object, conditioned on the parent being present.[1]

[1]It does not quite exactly represent this due to the fact that our model, in common with all work on capsules, only allows one of each kind of capsule to be present per instantiation.

In any instantiation, a child is part of a single object, which is represented with the selection variable s, which follows a categorical distribution:

$$s_j^k \,|\, \left\{t_i^{k-1}\right\}_{i=1}^{n_p} \sim \mathrm{Cat}\left(\frac{[\ldots, \rho_{ij}^k t_i^{k-1}, \ldots]}{\sum_i^n \rho_{ij}^k t_i^{k-1}}\right), 1 \leq j \leq n_k. \tag{4.3}$$

A capsule that is not on (i.e. $t_i^{k=1} = 0$) has zero probability[2] of being selected by a child. Note that we include a dummy parent capsule that does not correspond to an object, does not have any parents, and is always on. This allows capsules to not be a part of an object, having a small probability of being present but not explained by higher level objects, so "clutter" can be accounted for. This is also necessary in order to have a well defined probability if all parents are off.

The actual presence of the child, conditioned on its parent, is Bernoulli distributed and occurs with probability γ_{ij}^k:

$$t_j^k \mid \left\{s_j^k = i\right\} \sim \mathrm{Bern}\left(\gamma_{ij}^k\right), \qquad 1 \leq j \leq n_k \,. \tag{4.4}$$

If a child is on, its pose is given by the selected parent pose multiplied by a learned relative pose R_{ij}^k. By using a Gaussian likelihood with a learned variance c_{ij}^k, we allow for some mismatch between poses in individual images. If a child is switched off, it plays no part in image generation. However, we still represent the pose, just with a large variance λ_{off}.[3] This gives the conditional distributions:

$$A_j^k | s_j^k = i, t_i^k = 1, A_i^{k-1} \sim \mathcal{N}((I + A_j^{k-1}) R_{ij}^k, c_{ij}^k) \tag{4.5}$$

$$A_j^k | s_j^k = i, t_i^k = 0, A_i^{k-1} \sim \mathcal{N}(0, \lambda_{\text{off}} I))\,. \tag{4.6}$$

Notice we add the identity to A—this is primarily for convenience, as it lets us parameterise A as normally distributed variables centred around zero, rather than the identity.

When accounting for clutter by attaching to the dummy capsule, the child is given a random pose, conditionally independent of the other capsules. The dummy

[2] In practice, it is generally necessary to make this probability negligible instead of zero for numerical stability in log space

[3] It is necessary to do this because we will need to relax the discreteness of the parents to make inference tractable.

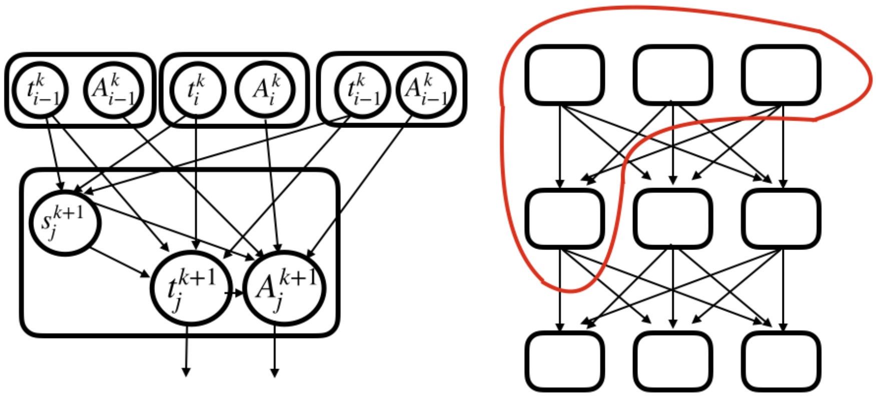


Figure 4.2: A sketch of the generative model for a capsule network. The broad connectivity pattern of the capsules is shown on the right, and the detailed graphical model between the random variables is shown on the left, for the region of the overall graph circled in red. Note the resemblance to a standard feedforward neural network.

capsule is given the zero index in each layer, and child capsules have a fixed small probability of attaching to it (implemented by $\rho^k_{0j} > 0$). A random pose is generated by setting a large c^k_{0j}, which is not learned.

Figure 4.2 shows a graphical representation of the conditional distributions between a capsule and the parent layer, as well as how capsules are arranged in a network.

So far, we have defined our model for relating the poses of complex objects to simpler ones. However, our final output is the pixels in images, not a collection of objects and poses. As a result, we need to relate the capsules in the final layer K (representing the simplest objects) directly to the pixels of the image. A simple way of doing this, following Kosiorek et al. (2019) and Tieleman (2014), is to associate a template T_j and alpha channel template α_j with every lowest level capsule. To build up an image, we transform the templates by their respective poses A^K_j, and then build up the image encoded by these using alpha compositing, drawing them in an arbitrary sequence, with the alpha channels weighted by the presence probability of that object.

$$\begin{aligned}
&\textbf{alpha_compositing}(T_{1:n}, A^K_{1:n}, t^K_{1:n}, \alpha_{1:n}) : \\
&\quad P \leftarrow \text{zeros} \\
&\quad \text{for } 1..n_K \\
&\quad P \leftarrow \text{over}(A^K_i T_k, t^K_i * A^K_i \alpha_i, P) \\
&\text{return}(P)
\end{aligned}$$

If we have two images C_a and C_b, with alpha channels α_a and α_b, then the over operation is

$$C_o = \frac{C_a \alpha_a + C_b \alpha_b (1 - \alpha_a)}{\alpha_o}$$
$$\alpha_o = \alpha_a + \alpha_b (1 - \alpha_a)$$

performed in parallel over each channel and pixel. This, in effect, draws C_a on top of C_b, accounting for the transparency of C_a encoded by the alpha channel. In order to have a data likelihood with full support in image space, we need to add noise to the data distribution. We use Gaussian noise, so the likelihood for an image X conditioned on the lowest level capsules is then

$$X_{ij} \mid t^K_{1:n}, A^K_{1:n} \sim \mathcal{N}(P_{ij}, \sigma)\,, \tag{4.7}$$

i.e. an independent Gaussian on each pixel centred on the mean of the composited image. Drawing the templates in this way has the side-effect of introducing an arbitrary ordering of the templates. This is somewhat undesirable, since it has no ‘physical’ basis, as the ordering of the templates is an internal detail of the model, but the template model is somewhat artificial in any case.

While our use of templates is shared by the recent approach by Kosiorek et al. (2019), we do believe this is a weak point in the current formulation of the model. Template rendering is a very inflexible rendering method compared to a neural generative model as used in the variational autoencoder (Kingma and Welling, 2013; Rezende et al., 2014), and it is reasonable to be sceptical that this would ever work for realistic images. However, using such a flexible rendering model would introduce non-identifiability between poses explained by the capsules, and by the neural network, as the variation in the appearance of the object due to variations in pose can also be explained by the neural component of the decoder. In order to focus on evaluating the object-part model and inference, we chose to follow the more interpretable existing template methods (Tieleman, 2014) to avoid these issues.

4.2.3 Inference

Image processing in our model consists of inferring the posterior over the latent variables conditioned on the image. Further, we need to perform inference in order to fit the generative model, as the global parameters of the generative process are not known in advance. The standard approach in deep generative modelling is to use variational inference with the reparameterisation trick, but this is complicated in this generative model by the presence of discrete random variables. However, due to the dependency structure of the model, the selection variables s are conditionally independent given all other variables in the model (see Figure 4.2). That is, given $A^k, t^k, A^{k-1}, t^{k-1}$, the s_i^k are all conditionally independent. As such, these can be marginalised out analytically for a cost in space and time of the order of the number of parent capsules in a layer.[4]

[4]This also avoids applying Jensen's inequality for these variables in the derivation of the variational bound.

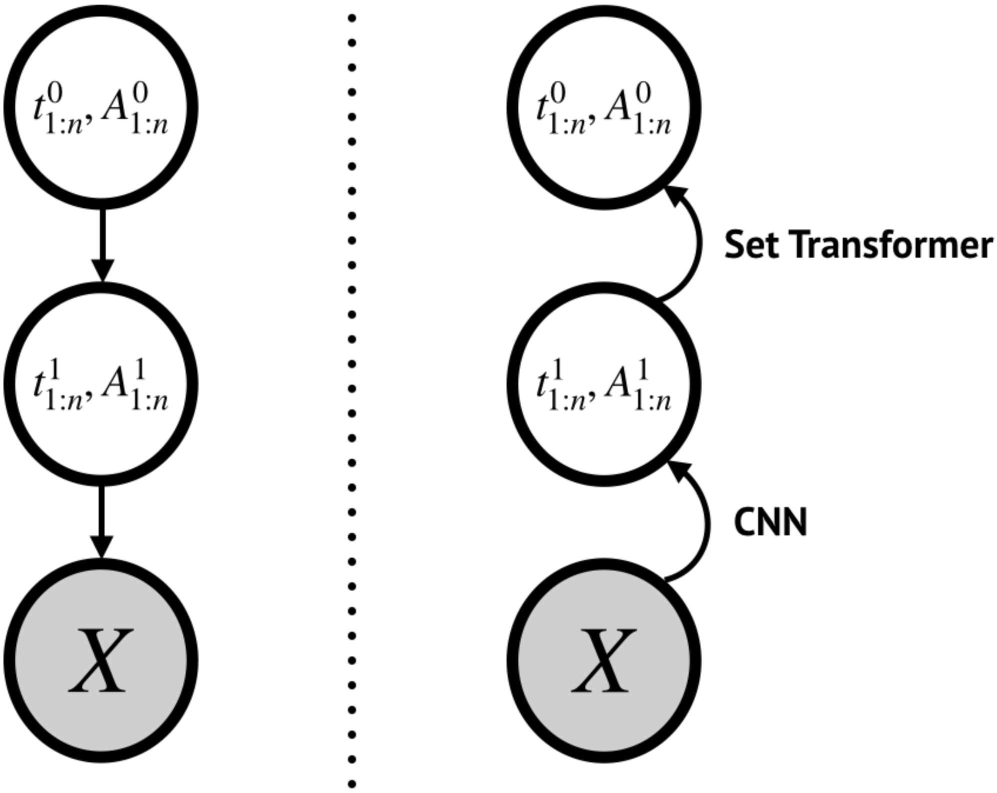


Figure 4.3: Left: the high level dependency structure in the generative model. Right: the dependency structure in the approximate posterior; variables are sampled from the approximate posterior at each layer before being passed on to the next layer, so the dependency structure of the amortised posterior is inverted compared to the generative setup (t^0, A^0 depend on t^1, A^1, rather than the other way around).

This cannot be done for the presence variables t, since the distribution of each t depends on *all* its potential parents $t_1, t_2, ...t_n$, and so enumeration would have a cost exponential in the number of parents in the model. Even for the relatively small numbers of capsules considered in this paper, this would be impractical, so instead we modify the model to relax the t variables, replacing them with concrete distributed variables (Jang et al., 2016; Maddison et al., 2017). With these changes, we can derive a tractable Monte Carlo estimator of the evidence lower bound for an approximating distribution over A and the relaxed t variables. In our inference network, we reverse the dependency structure of the generative model, as shown in Figure 4.3, parameterising $p(t^1, A^1)$ with a CNN, and $p(t^0, A^0)$ with a set transformer (more details on this are given in Section 4.4.) After enumeration of the s, the lower bound is then of the form

$$\begin{aligned}\mathcal{L}(\theta, \phi) &= \mathbb{E}_{q(t,A;\phi)}\left[\log p(t, A; \theta) - \log q(t, A; \phi)\right] \\ &= \mathbb{E}_{q(t,A;\phi)}\Big[\log(\sum_s [p(s \mid t, A)p(t, A, s)]) \\ &\qquad - \log q(t, A)\Big]\end{aligned} \tag{4.8}$$

A full derivation is included in appendix A.1. In our code, however, we do not need to use the full derivation of the bound; we implement our model in the probabilistic programming language Pyro (Bingham et al., 2018), which can derive an estimator of this ELBO automatically from the graphical model specification.

We wish to perform amortised inference (Kingma and Welling, 2013) for efficiency. This is partly for fast inference at test time, but mostly because, as in a VAE, we also want to fit the global parameters of the generative model by maximising the marginal likelihood. Using an expensive inference procedure would likely be too slow to make this practical, as the inference procedure has to be fast enough to run in an inner loop of this optimisation process.

4.2.4 Approximating distributions

In principle, we should be able to use any approximating distribution in our variational bound. In practice, training the model from initialisation with a full variational bound is extremely difficult. Since we cannot take expectations explicitly in the bound, we approximate it using sampling. The variance from this tends to mean that the model underfits if we train from this from the start, as near the start of training there is no real signal encouraging the variational posterior to be different from the prior, but if the variational parameters are sampled from the prior, there is also no signal leading to specialisation of the templates. In addition, the likelihood over the template pose is extremely strongly peaked, with a large likelihood for a 'correct' placement, but with large penalties for small deviations from this, which makes the problems arising from the variance much more severe.

In order to get around this problem, we use deterministic poses at least during the initial training of the model, leading to a partially variational bound, where the discrete variables are marginalised out, the presence variables are approximated with concrete distributions, and the poses are approximated with delta distributions. In other words, we perform maximum a posteriori inference over poses rather than finding a full variational approximation to their posterior, at least during training.

It is important to note that we verify that this is not a theoretical issue with the bound so much as it is a practical issue with training. Starting from a model initialised with the EM parameters, we are able to train an encoder with a full variational posterior, achieving comparable results to training with deltas. This achieves a tighter ELBO than training a model with the variational bound (with random sampling) from initialisation. Unless otherwise noted, however, we report results with delta poses in Section 4.3, as this is faster to train and we wanted to check a large variety of configurations, and we did not find that this re-training with the full ELBO had much effect on the metrics we use.

4.2.5 'Free form' variational inference

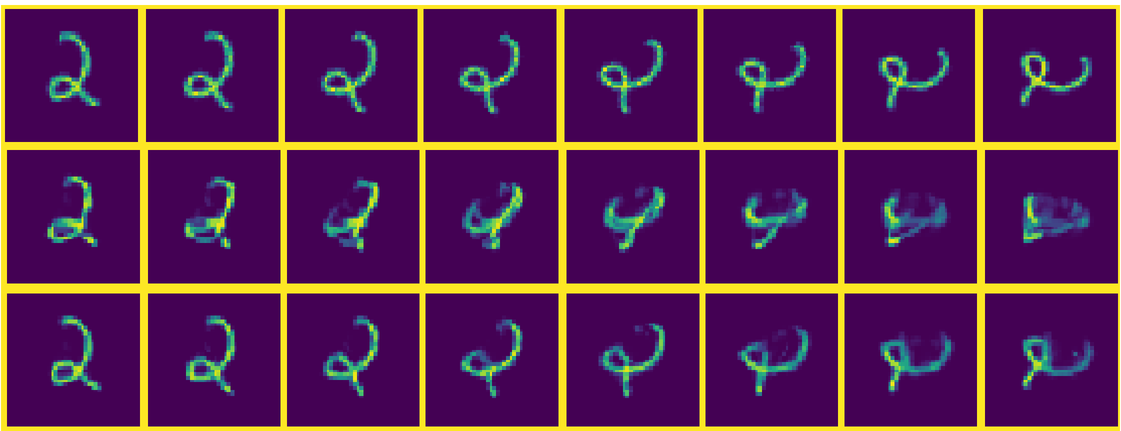


Figure 4.4: Top row: MNIST digit, with rotations applied. Middle row: Amortised reconstructions by a capsule model. Note the failure for some rotations. Bottom row: Reconstructions after free (test time) optimisation of our variational bound, showing that our unified objective is meaningful.

As well as amortised inference, it is possible to perform standard variational inference at test time using our objective. That is to say, rather than having the variational parameters ϕ be the outputs of a neural network, we instantiate a set of ϕ for each image, and optimise them separately using gradient descent. The analytic marginalisation of the s variables in our bound makes this correspond to an EM algorithm - we sample the continuous random variables from our variational distribution $q(\phi)$, then marginalise the discrete variables s conditioned on these analytically, and iterate these alternating steps.

This allows us to investigate the properties of our variational bound independently of our amortisation network. The advantages of this are demonstrated in Figure 4.4. This shows the results of reconstructing an out of distribution image, a rotated 2, using a generative model and inference network trained on standard MNIST. The encoder is able to generalise to small unseen rotations, but begins to fail for large rotations, which are very unlike the data it saw during training. However, if at test time we continue to optimise the variational parameters freely, initialising them at the values predicted by our encoder network, then the quality of reconstruction markedly improves, even for out of distribution images, as the *generative* model assigns a reasonable likelihood to these out of distribution images. We find also that optimising the bound at test time, initialising at the output of the inference network, before classifying based on the latent variables (see below) can improve classification accuracy based only on t^p by a modest amount. If we take a trained model and optimise the latent variables with respect to this bound, starting from the amortised model output, then we can improve the classification results based on t^0 quoted in Section 4.5 by a small amount, about 0.5%. The fact that we can do optimisation at test time and it improves the performance shows that our probabilistic bound is a useful and consistent objective. However, we do not report this in our results below, as it is extremely inefficient[5] to optimise this bound at test time using basic gradient descent relative to the forward pass of a neural network.

[5]This is because the gradient updates are slow to converge, so we need to take hundreds or thousands of steps to see real improvement in the bound, which means a significant amount of time spent per example, as each gradient step requires a pass through the generative model to calculate the gradient with respect to the variational parameters. If we take 1000 steps of gradient descent, which is not too unreasonable, then it takes 1000 times longer to process a single digit, which is hardly a fair comparison to other methods and would have made testing any sensible number of configurations extremely computationally expensive.

4.3 Related Work

There have been many different models which have been called 'capsules' over the years, with the idea dating back to at least Hinton et al. (2000). Interestingly, Hinton et al. (2000) used an explicit graphical modelling perspective, specifying a generative process over images, though not one that explicitly included poses. Similar hierarchical, graphical approaches were explored by Storkey and Williams (2003), though their model did not incorporate the pose offset (our R_{ij}^{k}) used in capsules. However, much subsequent work explored the ideas of having a vector activation in a purely discriminative model (with various training schemes), including explicitly geometrical (Hinton et al., 2011), and purely discriminative, but with an unusual routing mechanism (Hinton et al., 2018; Sabour et al., 2017).

Other routing approaches have been proposed, but we focus here on a probabilistic treatment of capsules, in particular based on Hinton et al. (2018), so we do not discuss these further. We focus on the view of capsules as representing objects specifically, so we do not consider the broader interpretation of capsules as meaning any model with vector valued activations which is sometimes used in the literature.

It is worth noting, however, that the routing algorithm of Hinton et al. (2018) is explicitly derived as an approximation to an EM procedure for a particular graphical model. The probabilistic model of a 'mixture of switchable transforming Gaussians' is not explicitly written down in Hinton et al. (2018) (its variational free energy is given). However, the graphical model between a set of parent poses and activations, and child poses and activations given in Section 4.2.2 corresponds closely to it. In (Hinton et al., 2018), however, this inference problem is used as a black box activation function in a model that is trained in a discriminative way. In contrast to the earlier routing algorithm of (Sabour et al., 2017), both steps of the routing algorithm of (Hinton et al., 2018) minimise the same objective function. However, this objective function is local to a layer, and as the overall algorithm is trained to minimise a discriminative objective, it has no particular relationship to the overall loss. In contrast, making the model generative means that the *entire model* minimises a *single*, consistent variational objective.

Approaches to capsules that have been trained discriminatively have often included a reconstruction term in the loss, for example, Sabour et al. (2017) and Rawlinson et al. (2018). In these works, however, the generative component of the model was a neural decoder that took the top-level capsule vectors as input but treated them as an unstructured latent space. In contrast, our generative model reflects the assumption that the latent variables in the model explicitly represent poses.

The decoder of the autoencoder model in the recent work of Kosiorek et al. (2019), is conceptually close to the generative model presented in this paper, and we use architectures based on the encoder described in their paper to perform inference. However, we adopt a more parsimonious approach with fewer templates and variables, contrary to Kosiorek et al. (2019) who use a more complex decoder—with more capsules, additional sparsity losses, and features like deformable templates and special features. Although these may allow for better performance, our main focus in this work is introducing a probabilistic treatment and analysing the model in this framework, rather than achieving state of the art performance. It is important to note that we find that the model described in Kosiorek et al. (2019) and our model suffers from a similar pathology when training on augmented data, which we describe in more detail in the next section.

Further, as mentioned before, the variables in the model used in Kosiorek et al. (2019) are not explicitly modelled as random, and there is no consistent joint probability defined over all variables in the model, despite descriptively referring to activations in the model as probabilities. The different components of the model being trained to maximise an image reconstruction likelihood and the higher-level part of the model maximising the log-likelihood of part poses, with a stop gradient between these two components (apart from on 'special features', which play an unclear role in generation). In contrast, we introduce a fully probabilistic view, which allows a clear separation of the *generative* modelling of capsule assumptions from the inference technique.

4.4 Model Architecture

In the following results, we use a relatively small model configuration of two layers, consisting of 16 higher-level capsules and 16 low level (template) capsules. We experimented with varying this but were unable to find a hyperparameter setting, which produced much better results, though we did not perform extensive tuning. We use a template size of 12. This means our generative model has very few learnable parameters compared to a neural model like a VAE decoder, with fewer than $30,000$ parameters. For comparison, even a simple VAE mapping from a 10-dimensional latent space to MNIST with 128 hidden units has over $100,000$ parameters.

For the CNN which predicts t^1, A^1 conditioned on the image, we use a 4 layer CNN, with a stride of 2 in the first two layers, and 32 channels. We use the attentive pooling mechanism described in Kosiorek et al. (2019). For our set transformer, which predicts t^0, A^0, we use 32 inducing points and 4 attention heads, with a hidden dimensionality of 128, across 4 ISAB layers followed by PMA (see Lee et al. (2019)). After the PMA layer, we use small MLPs to map the resultant 32 dimensional hidden state to the logits of the t^0 and the mean of the A^0 distribution (when using delta distributions in the variational/EM objective). Unlike in Kosiorek et al. (2019), there is no direct flow of information from the image to the higher level part of the model; we do not find this to be as important as reported in that work, possibly because we avoid the need for a stop gradient.

4.5 Experimental Results

4.5.1 Unsupervised classification

The specialisation of parent capsules can be evaluated by classifying inputs based on the presence variables t^0. We would expect that, if the generative model has been fit well, the top-level objects roughly correspond to the image classes in the dataset[6], and so we should be able to classify these with high accuracy based only on this latent representation.

[6] At least, we would expect this if the capsule assumptions are to be effective! In the next chapter, we will discuss some reasons why this does not in fact happen.

We also want to show that we can perform comparably to previous approaches on this task to validate our claim that our probabilistic formulation corresponds to the assumptions made in previous work on capsules. However, it is difficult to have an exact performance comparison due to conceptual differences in the interpretation of the model parameters. Rawlinson et al. (2018) perform linear classification on the latent vectors, but these use the latent representation of Sabour et al. (2017), which does not explicitly attempt to disentangle presence and pose. Kosiorek et al. (2019) perform linear classification and k-means clustering on metrics they refer to as the 'prior' and 'posterior' capsule presences, which do not correspond directly to posteriors over any variables in our formulation.[7]

In order to disambiguate these results, we report linear classification results based on the top-level activation t^0 only, as well as linear classification that also include the top-level poses A^0, which we find improves performance considerably, bringing our classification results in line with previous work. These results are shown in Table 4.1, along with results on generalising to out of distribution data described in the next section.

4.5.2 Generalisation to out of distribution data

In line with other work on capsules, we wish to show that our model generalises to out of distribution data. In previous work, the standard way to demonstrate this is to evaluate the ability to classify based on the latent representation of a model trained on a standard dataset on new viewpoints. Here, we use the standard benchmark of affNIST (Tieleman, n.d.). However, affNIST only includes relatively small changes in viewpoint, so we also train on more challenging out of distribution settings, namely MNIST rotated by up to 180 degrees.

Our affNIST results are better than reported in Rawlinson et al. (2018), and slightly worse than achieved in Kosiorek et al. (2019), but we do not use additional sparsity losses or deformable templates. These results are not state of the art,

[7] The the closest analogue would be that the 'prior' presence is roughly analogous to t^p, and the closest analogue to the 'posterior' capsule presence would be interpretable as the log of the expected number of child capsules that are coupled to a particular parent. This is discussed in more detail in appendix A.4.

	Rotated MNIST					affNIST
Max θ	0	45	90	135	180	-
t^0	0.96(0.01)	0.78(0.02)	0.56(0.02)	0.47(0.01)	0.44(0.01)	0.88(0.03)
t^0, A^0	0.97(0.00)	0.90(0.01)	0.77(0.01)	0.69(0.01)	0.65(0.01)	0.92(0.03)
VAE	0.96(0.002)	0.74(0.004)	0.47(0.004)	0.36(0.005)	0.33(0.005)	0.42(0.006)
Linear	0.93(0.00)	0.665(0.001)	0.43(0.0016)	0.33(0.002)	0.30(0.001)	0.31(0.002)

Table 4.1: Out of distribution performance: linear classification accuracy for model trained on MNIST and evaluated on rotated MNIST and affNIST. The values in parentheses are the standard deviations, computed over five runs.

but they validate our claim that our probabilistic model captures the capsule assumptions, as it shares their robustness to this setting.

4.5.3 Training on augmented data

Increased robustness to changes in viewpoint has always been a motivating example for capsule models. However, this has typically been assessed via experiments like those in the previous section, where a model trained on normal data is tested on augmented data.

Training only on data in a standard orientation, though, is a somewhat artificial scenario. In general, while we may want to augment the data with additional transformations to encourage robustness, like cropping and augmentation, we cannot necessarily know a priori what augmentations the dataset contains. If our model structure encourages robustness to *unseen* variations in pose, by explicitly constructing a graphical model that assigns a high likelihood to affine transformations of learned objects, then we might hope that the model will perform even better if we have access to these transformations during training. In practical scenarios, we may not have any choice about this - for example, if we are training on scenes of real objects, we will presumably have them presented in a variety of random orientations. As demonstrated in Figure 4.4, our amortised encoder does not necessarily perform well on unseen rotations, so we should expect this will help performance.

Somewhat paradoxically, we find that generative capsule models can perform *worse* in this scenario. We evaluate this by *training* models on rotated MNIST for increasing degrees of rotation. Despite the latent variables of models trained on normal MNIST being robust to rotations, as shown in the previous section, we find that increased rotations in the training set causes the model to fail to specialise its high-level objects to classes, as shown by the declining performance of classification based on the latent variables. These results are shown in Figure 4.5.

To verify that this is an issue with this kind of generative model more broadly, and not simply an issue unique to our model formulation, we also perform this experiment on the capsule autoencoder of Kosiorek et al. (2019), using their open source implementation. The performance of the two models are qualitatively similar, showing that this issue is not specific only to our formulation of capsules. In the next chapter, we investigate hypotheses for why this may happen, arguing that it likely stems from flaws in the assumptions made by the generative model, rather than any failure of our inference in particular.

4.6 Discussion & Conclusion

We have presented a probabilistic, generative formulation of capsule networks, encompassing many of the assumptions in previous work. Our model performs comparably to previous results in the literature on benchmarks they selected, validating that our model is at least a close approximation of these assumptions. Our probabilistic treatment makes the assumptions of the model explicit and provides a single, unified, consistent objective for learning the model, in contrast to previous work on capsules.

Capsule models aim to create explicit representations of objects and their poses. Unlike experimental evaluation in previous work, which has focused on benchmarking performance on tasks like unsupervised learning and generalisation to new viewpoints, we conduct detailed experiments into whether these theoretical properties are, in fact, preserved by this kind of model. Our results show that this kind of generative model may be helpful for enforcing desirable equivariance

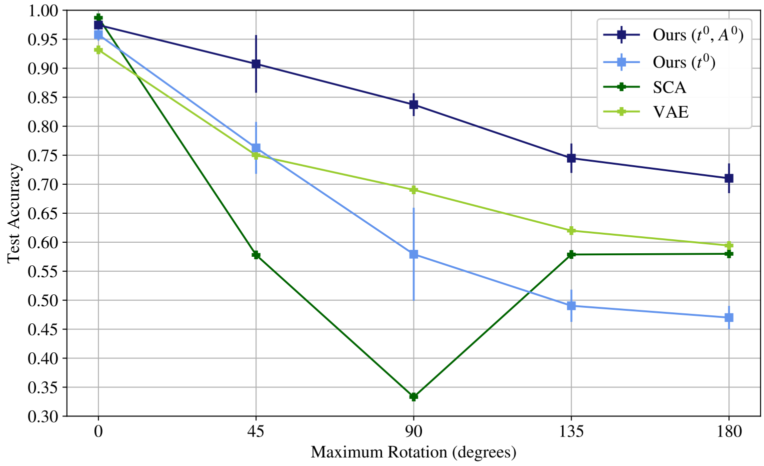


Figure 4.5: Classification accuracy of latent variables for model trained and tested on Rotated MNIST, compared to a Stacked Capsule Autoencoder (SCA) (Kosiorek et al., 2019). We evaluate the degree of specialisation by training a linear classifier on the poses and activations (for the capsule model). In this setting, neither our model using presences only or the model of (Kosiorek et al., 2019) reliably outperforms a simple VAE, in contrast to the results on *unseen* perturbations in Table 4.1. The results for our models are averaged over five runs. For SCA we use 24 templates as in the original paper; our choice of other hyperparameters is discussed in supplementary material.

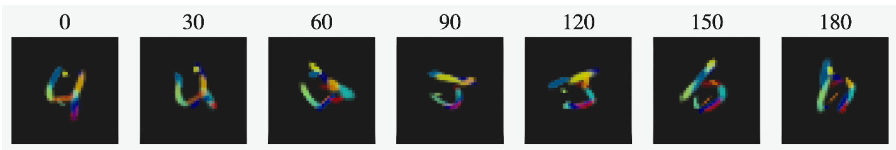


Figure 4.6: Inference of template arrangement as angular rotation increases. The figures shows the same input image rotated in increments of 30 degrees. Best shown on a computer screen.

Figure 4.7: Example learnt templates on MNIST

properties, but that this is far from sufficient. In addition, our results show that this formulation, while promising, is still in some ways underdetermined. The issues of identifiability of objects, in particular, suggest that changes to the generative model are likely necessary, possibly severely enough that the resultant model would no longer be recognisably the same as previous capsule models. In the following chapter, we will investigate the potential causes of this failure in more detail.

5

A Critical Analysis of Capsules

Contents

The aim of capsule models was to make computer vision more interpretable and robust by leveraging the geometrical structure of objects. In the previous chapter, we discussed a generative model which makes these assumptions explicit, but found several important problems with the implementation in practice, even in relatively simple situations. In this chapter, we analyse in more detail why this direction failed, and outline issues with capsules that models attempting to represent 'objects' explicitly should attempt to avoid.

5.1 Why Capsules?

As we have seen, capsule models specifically try to do two things that makes them distinct from other object focused approaches - they try to explicitly represent object *pose*, that is, geometrical orientation, and they try to represent the geometrical relationships between objects and their parts. In theory, this should lead to interpretable and efficient representations. We can think of an object like a bicycle consisting of parts - the wheels and the frame, for example - in a particular geometrical relationship to each other. Some of these parts occur in other contexts, meaning that our ability to represent these parts can be compositional; we can re-use our knowledge of recognising wheels when recognising other vehicles. But only when the right parts appear in the right orientation do we recognise it as a bicycle; a human would never classify a pile of wheels as a bicycle. In other deep models, there is some evidence that deep neural network units, while they can respond to fairly detailed patterns, are maximally activated by a picture that consists of many repeated instances of the pattern they search for (Olah et al., 2017). By explicitly building the model on the notion of objects as arrangements of parts, an important motivation for capsule models was avoiding this kind of behaviour (Hinton et al., 2018; Sabour et al., 2017).

Using the geometrical relationships between parts to recognise composite objects is also desirable because this geometrical relationship is invariant to the overall position of the object. This should allow robust generalisation to novel viewpoints, as the underlying relationship that we've captured doesn't depend on the angle we are looking at an object from. It has been argued that this should lead to improved robustness of capsule networks to changes in viewpoint or adversarial perturbation (Hinton et al., 2018; Qin et al., 2019).

5.2 A Critical Analysis of Capsules

In the previous chapter, however, we saw a number of experiments which challenged this narrative in sections 4.5.2 and 4.5.3. In particular, we found that while capsules generalise well to *small* transformations like Affnist, they do not generalise to arbitrary viewpoints very well. More worryingly, this remains the case even if the model is *trained* on augmented data, suggesting that the problem is not simply that our amortised inference fails to generalise out of distribution. We believe there are several potential causes of this failure, which are likely to overlap, and have various degrees of severity.

5.2.1 Identifiability of parts

A key part of the capsule motivation is that objects can be represented as arrangements of parts. What makes up a 'part' is not fixed in advance, but is supposed to be learnt from the data itself. In addition, there is no explicit restriction on how these parts are used by the model. The justifying intuition for capsules assumed that these parts would correspond to interpretable parts of objects, like wheels or handles, but there is nothing in the model that enforces this.

We can gain some insight into what the capsule model actually does by visualising the way that the parts are being used to reconstruct the images, showing each template of our learned model (each part capsule) in a different colour. Ideally, we want pose of the parts of the model to change as the pose of the object changes, while keeping their relative orientation the same. This is what would happen if our model behaves according to the motivating picture above - a transformed scene will be represented by correspondingly transformed parts, so that the relationships between the parts making up the object remain constant.

Instead, the behaviour we observe in our model is visualised in Figure 5.1.

As this figure shows, rather than translating parts, the model tends to represent transformed versions of an input by using different templates for different parts of the image. This is particularly visible on the green template making up the stroke of the 2; rather than transforming this part, the model keeps it in more or less the same position as the input image is rotated.

A key assumption when we were describing capsules is that the parts were identifiable - a translated object would *have* to be represented by a set of translated

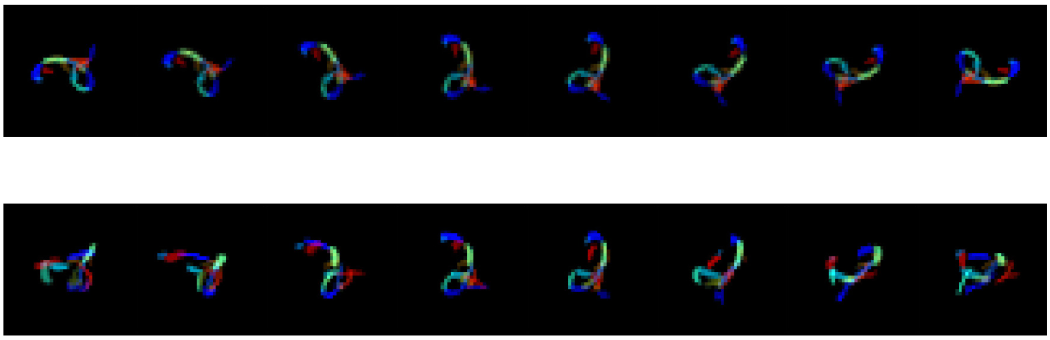


Figure 5.1: Top row: ideal reconstructions, for an arbitrary decomposition into parts. As the image rotates, according to the motivating narrative of capsules, the parts of the image should rotate in a corresponding way. The row below are the actual reconstructions of the model when presented with rotated input, demonstrating that the model does not represent the transformed image as a set of correspondingly transformed parts, but uses a different arrangement of the parts for each transformed input.

parts, or at least this would provide some representational advantage, such that the model would prefer it. But it's clear that this isn't the case - here, when objects rotate, instead of shifting the corresponding parts, the model simply uses different part capsules to represent a given region of the image when the image is rotated.

These reconstructions are actually reasonable in terms of the image likelihood, but this violates the assumptions we depended on to describe the representational advantage of using capsules - the same object capsule cannot be used to describe the 2 in the rotated images above, because the geometrical relationship between the parts is different in different orientations, rather than being invariant. The fact this happens shows that a key assumption of capsules - that transformations of objects will be represented by corresponding transformations of all their parts - is not true in this implementation.

This seems to happen, at least in part, because of the assumption that parts occur only once in an image. However, as we see in these images, the parts the model has converged to are in fact fairly similar, and largely interchangeable. One could formulate a generative model which could avoid the issue of repeatable parts, by allowing objects to appear a variable number of times, though inference would likely be much more difficult.[1] However, the issue of parts potentially converging to be very similar would remain; this comes from the fact that capsules encode object identity as being *discrete* (indexed by location i in the model architecture). As such, if two parts converge to be similar, there is no real way for the model to recognise that these parts can be interchanged. An alternative could be parts and wholes with a 'continuous identity'; for instance, to have parent and child identity represented by a continuous vector z, and have their poses represented as $R(z^p, z^c)$. While this could be interesting, a model like this could not have parts vote for wholes, as the part-whole relationship matrices would depend on the identity of the parents. This would not be known at inference time, resulting in a model which would have to give up on the straightforward feed forward inference pattern allowed by the assumptions of the capsule generative model. In addition, this creates a further identifiability problem between changes in appearance being explained by changes in the pose of the parent object and changes in the *identity* of the parent. So such a model, based on a hierarchy of latent vectors, could be interesting, but would arguably have to give up on the most distinctive feature of capsules, explicit modelling of pose.

5.2.2 Identifiability of pose

A related issue is the non-identifiability of object pose. If an object has internal symmetry, its pose may only be defined up to a transformation by the symmetry group of the object. For example, a square is invariant to 90 degree rotations, and so its rotational angle can only be defined modulo 90 from an image. This is important because an unspoken assumption in the motivating picture for capsules is that the pose of the parts are obtainable and unambiguous. Failure to take this into account can lead to issues when dealing with symmetrical (or approximately symmetrical) templates, as shown in the Figure 5.2. This is an issue with the generative model - several solutions which lead to exactly the same image have different likelihoods under the generative model due to this unobservable component

[1] Allowing repeatable parts means that the number of instances of an object is variable. Allowing this is possible, as in the generative model of Attend, Infer, Repeat (Eslami et al., 2016), but complicates inference considerably, as the inference algorithm has to break the resultant symmetry between part random variables in some way, such as by using an iterative inference algorithm.

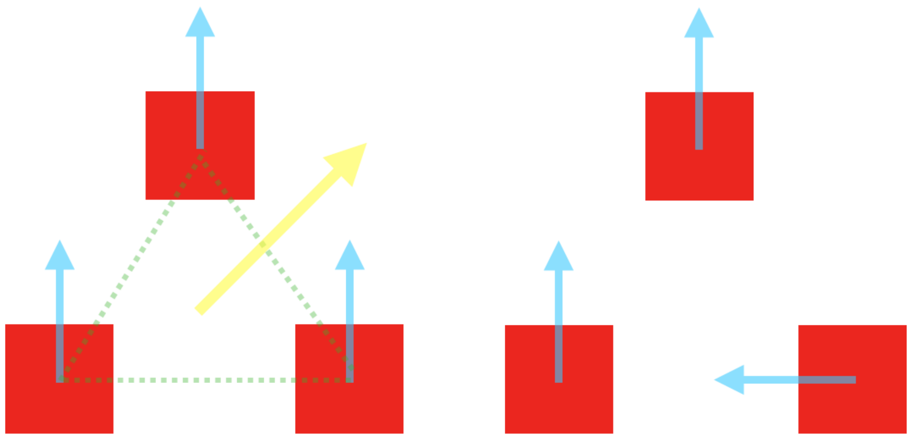


Figure 5.2: In this figure, we see the poses of three squares (represented as blue arrows) and the pose of the triangle, in yellow, which is an arrangement of the three squares as transformations of the triangle pose. The image on the right cannot be represented using the triangle object if we deal with full poses, even though the images are identical, because the 'hidden' pose of one of the squares is different.

to the pose variables. In principle, a sufficiently powerful inference procedure would be able to work around this by representing a multimodal posterior, but it is less clear that an amortised model can avoid this issue.

We can see this problem in an (even more!) simplified setting. Consider a template model with only a single learnable template, with a single pose, and a dataset consisting of a *single* MNIST image with random rotations applied to it. The inference network must learn to predict the rotation angle θ of the template, while the template must be learned along with it. In principle this is a simple task that can be solved perfectly by this model, as the network can achieve perfect or near-perfect scores simply by making the template match the base image, then learning to output its pose correctly, which could be achieved by memorisation. Interestingly, however, we find that we are unable to get a standard convnet to solve this problem, as shown in Figure 5.3.

The templates learned in practice on MNIST, at least for our model, suffer from this issue - many have (at least approximate) internal symmetry. Even worse, many templates are extremely similar. This leads to another identifiability issue. Consider two templates a and b, with interchangeable appearance. The *internal*

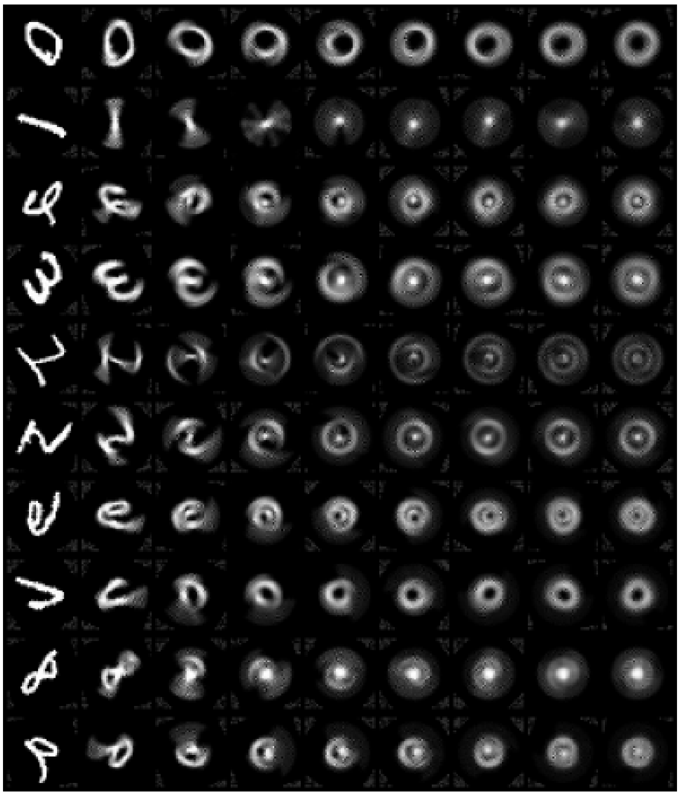


Figure 5.3: The templates learned by a single part capsule trained on a dataset consisting of rotated versions of a single image, for every digit class in MNIST, for a dataset consisting of increasing degrees of rotation, 0 on the left and 180 on the right. That is, each cell is the template learned on a dataset only consisting of the leftmost image augmented by an increasing amount of rotation. We notice that the model has a tendency to respond to variation in position by learning averaged templates which remove any training signal for the pose inference.

variables in the model which govern their relationship to other objects (the R_{ij} and ρ_{ij} parameters) are indexed by *position* in the model (the index of the capsule). This means that if these two templates are exchanged (by swapping their pose and presence parameters) the image described by the model may remain the same, but the likelihood assigned to this arrangement by the object capsules may be different. This leads to a similar problem of hidden differences in the internal details of the model that have no effect on the image likelihood.

5.2.3 What is an "object" really?

The problems with the identifiability of objects and poses are difficult, but could perhaps be resolved technically. However, we think there is also a conceptual issue. Papers on capsules, and indeed on object oriented learning more broadly, often talk as though it is obvious exactly what an object *is*;

> A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part (Sabour et al., 2017)
>
> Objects are composed of a set of geometrically organised parts (Kosiorek et al., 2019)
>
> As the viewpoint changes, the pose matrices of the parts and the whole will change in a coordinated way (Hinton et al., 2018)

But while an ‘object’ or a ‘visual entity’ may seem to be an obvious or intuitive concept, in reality it is anything but. As the quotation at the beginning of this part of the thesis points out, the division of the world into objects, and these objects into constituent parts, is clearly not an absolute property of reality, but a conceptual division. For example, it may seem obvious that an everyday object, say a bicycle, can be divided into simple parts - frames, wheels and so on. But these distinctions are not absolute - these components will also be composite in some sense (wheels are made of spokes, tyres etc.). Are these also parts? What about a person riding a bike? Is the person an object, or a part of a bigger cyclist object?

We might be tempted to think that we can get around these problems by using hierarchical models, but there is still a considerable problem of how to organise the hierarchy - for instance, if we assume (as capsule models do) that the hierarchy can be naturally organised into levels, with high level objects consisting only of the levels below, then this becomes problematic quickly. To stick with the same example, what are the objects and parts in an image of a person standing next to a person on a bike? For a hierarchical object model to make sense, this scheme should have two independent objects; ‘person’ and ’person on bike’. But if ‘person on bike’ is represented as an arrangement of lower level ‘person’ and ‘bike’ objects, 'person' can't *also* be a top-level scene object unless it is redundantly duplicated at multiple levels, which would mean there is no compositional advantage to an object-focused scheme. On the other hand, if we represent this instead as three high level objects (person 1, person 2, bike) then in a capsule type scheme we must treat the poses of the person on the bike and the bike as independent or quasi-independent. This is clearly a nonsensical representation of the underlying physical relationship, but is a restriction imposed by the simplifications in the capsule model of how objects and parts are related.

There is clearly a more philosophical problem regarding the breakdown of a scene into atomic objects. This decomposition is task dependent, rather than being absolute. For instance, how is a paragraph of text broken down into visual objects? It may seem that this works well in a capsule view - at the base level we have the strokes that make up letters, then letters themselves, then words, then sentences and so on. But this view ignores the fact that people are quite capable of recognising letters with miss-spelled words, which is difficult to explain if words are recognised purely as geometrical arrangements of letters, and it also implies that compound words cannot re-use any knowledge (i.e draw and drawbridge must be represented as totally distinct entities, without composition). It's clear that the decomposition of a scene into objects is somewhat task dependent.

But by focusing a capsule model on reconstruction, we further make the assumption that *visual* objects, which are useful for reconstructing images, will also be useful for other tasks we are interested in. As we see in Figure 5.1, the capsule model learns to explain images as arrangements of stroke-like 'parts'. But it's not obvious that these parts are useful or meaningful for anything other than reconstructing MNIST digits - a decomposition into parts useful for accurate image reconstruction is not necessarily useful for anything else.

Representing the poses of objects explicitly may be useful, for instance, for a robotics system trying to manipulate the world, where *which* objects should be represented in this way is clearly dependent on the specific task. The relevant definition could either be enforced by additional losses, partial supervision, or the structure of the loss.

This discussion seems rather too abstract and philosophical for a machine learning thesis. It's relevant since capsule networks, by design, attempt to embed a single, fixed, decomposition into parts and objects into the structure of the network itself, which requires making restrictive choices about how these 'objects' relate to one another which, as outlined above, will almost certainly be wrong. Even if the identifiability issues with the generative model could be resolved, it is not clear that this is a good idea, as it forces us to concern ourselves with the issues outlined above.

5.3 Conclusion

Drawing definite negative conclusions from empirical work is difficult. It is possible that our disappointing results were simply missing some trick which would solve some of the issues described here. However, we think that we have outlined some reasons, backed both by theoretical discussion and experiment, to be sceptical of this kind of model going forward. We think the value of formulating a generative model formally, as we did in the previous chapter, is that it allows these assumptions to be made very explicit, allowing them to be critiqued and examined in detail.

Here we summarise our own conclusions, as well as briefly discuss some parallel developments in capsules that have been published in parallel or subsequently to the work that we build on in this chapter

5.3.1 Pose inference

As our results on single capsules and training on augmented data show, performing pose inference is extremely difficult for standard neural networks. As a result, networks with a pose inference component often need to be tightly regularised in order to force them to use this component to represent images. We find this is not always possible even in a vastly simplified setup. It's possible that using networks with more powerfully equivariant designs in the inference network for the parts could impart a more powerful inductive bias that would enable this to be learned more efficiently, but we are not aware of any demonstrations of this.

5.3.2 You need to know what you want to have objects for

As we discussed above, while objects and parts are intuitive concepts, they are also vague. The decomposition of a scene into objects and the objects into parts is not really a property of reality, but a feature of our models. Capsules *implicitly* make very strong assumptions about what a part 'is' and how parts are related to one another to form objects - in effect, introducing an operational definition of object which may not map onto anything we want to use the 'objects' for (classifying images, manipulating the physical world, etc.).

5.3.3 Related work and future directions

There are many models which have been described as capsule networks, many of which are substantially different to the kind of model we discuss in this chapter, though the motivation of capturing the geometrical relationship between objects and parts via explicit representation in the model is a common feature to all. Some of the criticisms in this chapter apply mainly to capsule models dealing with image data, the domain for which they were originally proposed. Some models dealing with point cloud data have shown more promise (Srivastava et al., 2019; Sun et al., 2020; Zhao et al., 2020). Point clouds are an interesting domain for modelling objects using poses since they avoid to a degree the problems of defining the primitive objects and inferring their poses - a point is the ultimate 'primitive object', and already has a 'pose' in the form of its location vector. Because of this, point clouds can be readily transformed by applying random transformations to them to generate different 'viewpoints' of the same object, a mechanism which both of these methods exploit in order to generate a training signal for pose inference. The underlying issues around the usefulness of a part-whole decomposition still apply to this kind of model.

Recently, there has been more interest in so called 'slot' models, such as those described in Burgess et al. (2019), Eslami et al. (2016), Greff et al. (2019), Kipf

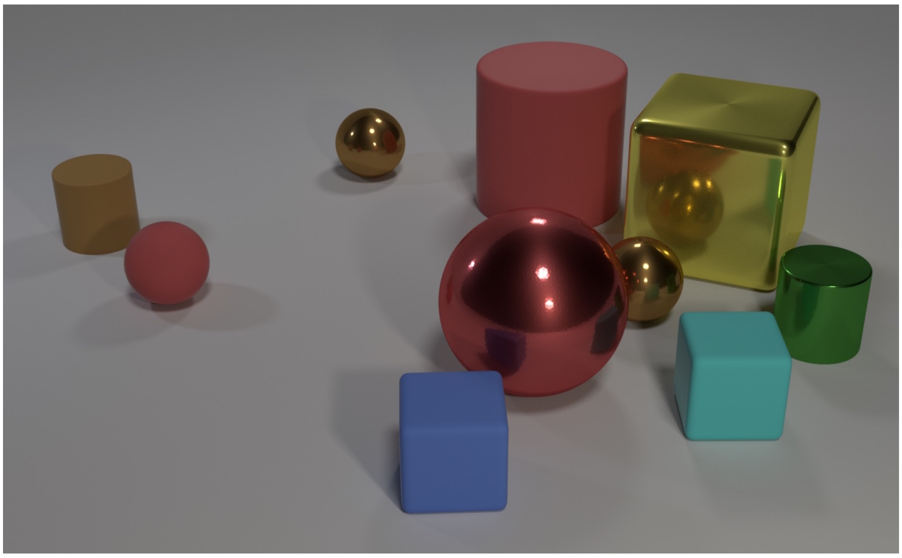


Figure 5.4: An example image from the CLEVR dataset (Johnson et al., 2017), frequently used to test 'object focused' models. While it is of course sensible to test ideas like object-focussed models on simplified settings like this, the definition of object is so self evident in datasets like this that some of the issues around what objects mean are obscured

et al. (2019), and Locatello et al. (2020). As discussed, one drawback of capsules is that object identity is encoded by position *in the model*, that is, by the index i of the capsule. If we permute the pose and presence of capsule i and j in our model, we have changed the likelihood of the data, unless we also swap all the variables associated with the edges in the graphical model, like the affinities and relative poses R. In contrast, so called slot models specify an object structure where a scene is described by a set of latent variables $\{z_i\}$, such that the model is invariant to permutations $\{z_{\pi(i)}\}$ of the object latents. Slot models do not introduce any hierarchy between objects, and they do not explicitly model pose, which avoids many of the issues mentioned above. While some care needs to be taken, in our opinion, about the definition of 'object' - slot models have mostly been tested on very simple datasets such as CLEVR (Johnson et al., 2017) where the possible difficulty of defining objects is avoided - this is possibly a more promising direction if modelling objects is relevant to the desired use of the model, though there is no reason to suppose that slot based models would be more robust to geometric transformations.

A combination of modelling an object hierarchy with exchangeable object slots could be an interesting direction for future work, though the caveats outlined in this chapter about the validity of the object-part hierarchy could still apply. But this objection only really applies to attempts to discover 'objectness' in an unsupervised way - the fact that an object decomposition is arbitrary is not really relevant if it is a useful inductive bias for a specific task. Object structured models may yet prove to be useful in this regard, given an operational definition for which objects are relevant.

However, there is no particular reason to suppose that these object structured models would be more useful for *uncertainty* than standard neural networks, as far as we can see. In contrast, capsules had a compelling narrative that they ought to have better uncertainty out of distribution by reflecting the underlying geometry of the problem, which made them a direction worth exploring in attempting to answer the central question of this thesis. However, for the reasons outlined in this chapter, we ultimately don't think that capsules are a promising alternative to standard Bayesian neural networks as a method for obtaining more useful quantifications of uncertainty, as both the practical implementation and the underlying motivation for the model suffers from serious problems.

Part III

Controlling Uncertainty

A neural network sounds like something that will solve intelligence, but a Gaussian process sounds like something to do with making fertiliser

Michael A. Osborne

6
Bilipschitz Models

Contents

6.1 Introduction

Our work on capsules was motivated by the possibility of better out-of-distribution behaviour, as models which reflect the invariances of a dataset ought to generalise better. Including invariances in neural networks remains, in our view, a promising research direction in general, but as outlined above we do not feel that capsules were successful in resolving this. With this in mind, we move to presenting a family of models with a more limited goal; to enforce the return to a controlled prior away from the training data. As we saw in Chapter 3, one way to achieve this would be ideal Bayesian inference, which can be achieved by ensembling large numbers of samples from an MCMC procedure like HMC. However, this is not particularly practical due to the computational cost of HMC. In this part of the thesis, we aim to reproduce some of the properties of idealised HMC inference in terms of returning to a prior using models with practical computational costs, in particular aiming at models which only require a single forward pass through the deep part of the model.

Until recently, the standard approach for obtaining uncertainty estimates in deep models relied on various forms of ensembling: either Bayesian methods such as various forms of variational inference(Blundell et al., 2015; Gal, 2016; Graves, 2011; Hinton and van Camp, 1993), Laplace approximations, various forms of Markov Chain Monte Carlo inference (Neal, 1995),which estimate an average over an approximate posterior by random sampling, or non-Bayesian ensembling methods which simply use a randomised ensemble to estimate the uncertainty of a network, as in Lakshminarayanan et al. (2017). This approach is often inadequate: neural networks have very high dimensional feature spaces, and so averaging over them with a reasonable number of samples is fairly difficult. In addition ensembling is also significantly more expensive; even the most basic ensembling approach of training a model k times from multiple seeds is naively k times more expensive to train and requires k times more operations at test time.

The behaviour of models out of distribution is not always a concern in practice, but there are several applications where it would be desirable to quantify model uncertainty out of distribution in a reliable fashion. For instance, consider an astronomical classification pipeline, where we wish to automatically classify well known types of (say) galaxies into categories automatically, and refer particularly difficult or unusual examples to a human observer. In this kind of application, the fact that neural networks often display high confidence on out of distribution images is an obvious disadvantage.

One potential solution to this problem is to replace the softmax layer in a neural network with a distance-sensitive classification method, which makes predictions by comparing input points to some set of support vectors using the Euclidean distance. By using a method like this, one guarantees that the model will revert to a default prior for points with feature vectors far from these support points.[1] This kind of model includes kernel machines, such as (exact or approximate) Gaussian Processes or Support Vector Machines on top of deep networks, as in deep kernel learning (Huang et al., 2015; Wilson et al., 2016b; Yang et al., 2015) as well as methods based on prototypical cluster points in the feature space of a neural network to represent classes, such as RBF networks (LeCun et al., 1998b) or the prototypical networks proposed in Snell et al. (2017).

However, while this solution has been proposed many times, it has not been widely adopted. Such methods are frequently difficult to train. Partly this is due to the difficulty of scaling most kernel machines to the large data regime; much of the literature on this kind of model is based around addressing this problem. In addition to this, however, the prior-reverting properties of this kind of distance sensitive model in input space are frequently not preserved if they are instead applied to the feature space of a neural network, as the neural network is able to manipulate the input points arbitrarily (Ober et al., 2021). In particular, this kind of model is prone to *feature collapse*, where a set of input points is collapsed onto a single point in feature space, which means that the guarantees of the distance-sensitive model become meaningless.

It's natural to assume that the possibility of feature collapse comes, in part, from the flexibility and power of deep networks as feature extractors. Therefore, to address this it is natural to try to constrain the flexibility of these models, in order to preserve the usefulness of the distance-aware final layer. This section of the thesis introduces a regularisation scheme which addresses this problem, describes some empirical results obtained using it, demonstrating uncertainty out of distribution which compares favourably to deep ensembles, and then discusses theoretical explanations for why the scheme is effective in more detail.

[1] At least for stationary kernels. Technically, it is possible to use kernels which do not have this property, such as a linear or polynomial kernel

6.2 The Bilipschitz Constraint

We want a regularisation scheme in order to avoid or control feature collapse in deep models. A natural way to do this is to enforce a *bilipschitz* constraint. A function $f : X \to Y$ is bilipschitz if it has both an upper and lower Lipschitz bound: that is, a bound of the form

$$L_1 d_X(x_1, x_2) \leq d_Y\left(f(x_1), f(x_2)\right) \leq L_2 d_X(x_1, x_2), \forall x_1, x_2 \tag{6.1}$$

where $d_{X,Y}$ are distance functions defined on X and Y respectively, and L_1, L_2 are positive constants. If we assume that $Y = f(X)$, i.e that f is surjective, then this is equivalent to enforcing that both f is Lipschitz, and that its inverse function f^{-1} exists and is Lipschitz continuous. If f is not surjective, then the inverse cannot be defined everywhere, as then there exist points $y \in Y$ with no solution to the equation $y = f(x)$, but we can get around this situation by restricting the definition of the inverse to the image of $f(X)$.

The existence of a well defined inverse function necessarily prevents feature collapse; if x, y are distinct points then their images $f(x), f(y)$ must also be distinct. The lower Lipschitz bound also controls how close two distinct points can be mapped together by f in terms of their distance in X. A bilipschitz bound can be enforced on a parameterised mapping by using a residual connection, $f(x) = x + g(x)$, then enforcing an upper Lipschitz constraint on the arbitrary function g, which can be done using spectral normalisation, as described in Section 6.2.1.

It is straightforward to demonstrate why this construction enforces a bilipschitz bound. The upper Lipschitz bound holds because Lipschitz bounds are additive, so if $\mathrm{Lip}(g) = L$, then $\mathrm{Lip}(f) = 1 + L$. The lower bound holds because (Behrmann et al., 2019)

$$\begin{aligned}
||f(x) - f(y)|| &= ||x + g(x) - y - g(y)|| \\
&= ||(x - y) - (g(y) - g(x))|| \\
&\geq |||(x - y)|| - ||g(y) - g(x)||| && \text{Reverse triangle inequality} \\
&\geq |||(x - y)|| - L||x - y||| = (1 - L)||x - y|| && \text{Lipschitz bound on } g
\end{aligned}$$

Note that f is enforced to be bilipschitz only if $L < 1$, otherwise the last bound is trivial.

Another important observation about the above argument is that it only holds for dimensionality preserving mappings, i.e when the input and output dimensions of f are the same, otherwise the expression $x + g(x)$ is not well defined. Nevertheless, this spectral normalisation scheme has been applied to residual mappings which include dimensionality reduction, which should be regarded as a regularisation scheme motivated by the above argument but not strictly justified theoretically by it. Despite this, we frequently refer to the regularisation scheme of residual connections and spectral normalisation as bilipschitz regularisation to reflect this motivation and to accord with use in published literature, despite it being technically inaccurate. This is discussed in depth in Section 8.2.2.

6.2.1 Implementing spectral normalisation

The above regularisation scheme assumes that it is possible to impose an upper Lipschitz bound on an arbitrary mapping g. There are a variety of ways to achieve this which are fairly compatible with standard deep learning architectures. A Lipschitz bound on a network can be implemented by bounding individual network blocks because these bounds are multiplicative: that is, if f_1, f_2 have Lipschitz bounds L_1, L_2, then $f_1 \cdot f_2$ has Lipschitz bound $L_1 L_2$.

Most commonly used activation functions are Lipschitz, and as they are typically fixed scalar functions applied pointwise it is fairly trivial to calculate these bounds analytically by calculating $\max_x \frac{d}{dx}\sigma(x)$. The Lipschitz bound of many common activation functions is 1; this includes the ReLU and tanh activations.

Most learnable layers of interest – linear layers and convolution layers[2] – are linear operations, and so their Lipschitz bound is equal to the largest singular

[2]The self attention layer, as mentioned in Section 1.2.2, is an increasingly important operation in modern deep learning, but the work presented in this section of the thesis is mostly concerned with standard convolution networks, though applying these ideas to attention models could be an interesting future direction. Self attention is *not* Lipschitz bounded, as shown in Kim et al. (2021), though that paper does propose a Lipschitz-bounded alternative to self attention with similar performance.

value of the linear operator [3]. For linear layers this is easily computed, either exactly (by taking the singular value decomposition) or by power iteration (Gouk et al., 2021; Miyato et al., 2018). The power iteration method can be adapted to work on convolution layers (Gouk et al., 2021). The exact spectral radius of a convolution layer can be computed in the frequency domain (Sedghi et al., 2018), similarly to directly computing the SVD of a linear layer. In our experiments in chapters 7 and 8, we found that the power iteration technique was generally the most practical, despite not being exact, as it is much faster to compute and we did not find that switching to the exact method had a significant effect on training accuracy. As a result, this is what we mean throughout this thesis where we refer to 'spectral normalisation' unless otherwise specified.

Other frequently used components of deep networks have non-trivial Lipschitz constants. An important component, whose Lipschitz bound is occasionally neglected, is *batch normalisation*. BatchNorm transforms the input as follows:

$$x_{out} = \mathrm{diag}\left(\frac{\gamma}{\sqrt{\mathrm{Var}(x)}}\right)(x - \mathbb{E}[x]) + \beta, \tag{6.2}$$

where γ and β are learnable scale and shift parameters, and $\mathbb{E}[x]$ and $\mathrm{Var}(x)$ are online estimates of the mean and variance of the input respectively. It has a Lipschitz constant of $\max_i |\frac{\gamma_i}{\sqrt{\mathrm{Var}(x)}_i}|$ (Gouk et al., 2021). As a result, one can easily define a Lipschitz bounded version of BatchNorm by dividing through by this constant if it is larger than a given value, though this obviously affects the ability of BatchNorm to normalise its input to a constant standard deviation.[4]

6.2.2 Gradient penalties

Earlier work on this family of models, in Amersfoort, **Smith**, Teh, and Gal (2020), used a gradient penalty instead of a spectral constraint. This model used c centroids e_c, with corresponding projections W_c, to represent the classes of the problem, assigning a point to the class of its closest centroid, and using the RBF distance

[3] Assuming that the distance metric we care about is Euclidean distance

[4] Specifically, it makes it impossible for BatchNorm to increase the scale of activations with a small standard deviation, as this would have a high Lipschitz constant.

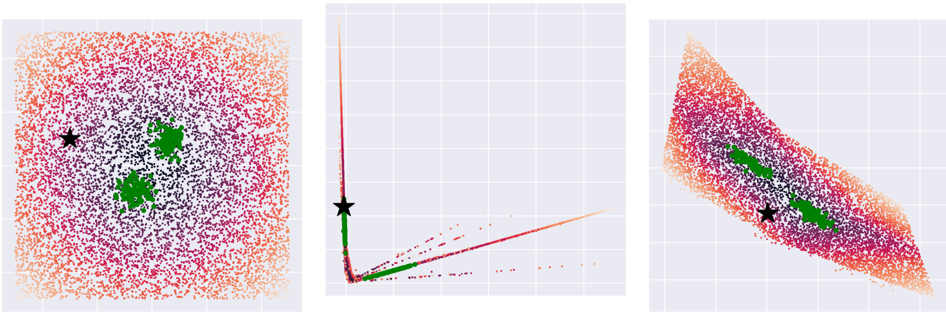
to the closest centroid as a measure of the certainty of that assignment.[5] Letting $K_c(x) \propto \exp\left((W_c f(x) - e_c)^2\right)$, the method used a gradient penalty of the form

$$\left[||\nabla_x \sum_c K_c(x)||_2^2 - 1\right]^2 \tag{6.3}$$

Intuitively, the effect of this penalty is to ensure that there is always some first-order sensitivity to x in at least one of the centroids. The aim of this penalty was to counteract feature collapse. This penalty has a relationship to bilipschitzness; the Lipschitz constant of a differentiable function is the same as the maximum operator norm of its Jacobian. Similarly, the lower Lipschitz constant of a bilipschitz function is equal to the minimum over x of lowest singular value of its Jacobian.

However, this intuitive justification is not particularly rigorous, and this constraint is also subject to concerns about dimensionality discussed in more detail in Section 8.2.2. Briefly, since the vector $K_c(x)$ has dimensionality c, its Jacobian has dimensionality $c \times n$ where n is the number of input features. As a result, the matrix must have a null space of dimension $n - c$, so there must be zero sensitivity along a space of this dimensionality at every point in input space. This implies that the lower Lipschitz bound is always zero, which means the connection to invertibility is less than obvious. However, we found in Amersfoort, **Smith**, Teh, and Gal (2020) that this penalty was effective empirically, and that the lower bound on the gradient was extremely important for performance. In practice, spectral norm as used in Amersfoort, **Smith**, Jesson, Key, and Gal (2021) and Liu et al. (2020) is probably preferable, as the gradient penalty is expensive to implement (requiring double backpropagation) and can lead to somewhat unstable training. On the other hand, a gradient penalty does allow more flexibility in the choice of architecture as it doesn't require residual connections, which could be useful in the future.

[5] This is very much not a Bayesian procedure!



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

Figure 6.1: 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) is used to visualise where points in input space are mapped to in feature space. 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 measures based on the feature space distances insensitive to changes along the compressed dimensions. In particular, the star moves from an unrelated area in input space on top of class data in feature space, despite being in a low-probability region. In contrast, the constrained mapping only distorts the relative distances of the other points in a controlled way.

6.3 Sensitivity and Classification

It is worth briefly addressing why these penalties are necessary; why are classification models prone to feature collapse? It's important to realise that feature collapse is not necessarily a bad thing from the perspective of minimising the loss. For instance, consider a contrived classification problem with two inputs, x_1, x_2. Assume both are unit Gaussian noise, and assume that the label y depends only on one of these variables: say $y = \text{sign}(x_1)$. In terms of minimising the loss, the best choice is for a feature extractor is to simply throw away x_2 and map the data onto a plane defined by x_1. An empirical demonstration of this kind of scenario is shown in Figure 6.1.

This behaviour can even be directly encouraged by the loss; distance sensitive methods like centroids in feature space directly minimise the loss if every instance of a class can be mapped to a single point in feature space, and methods based on a linear decision boundary minimise their loss if points can be mapped as far from this boundary as possible.

Using Bayesian models, like Gaussian processes, in feature space also suffers from this issue. As discussed in Ober et al. (2021), the ELBO for a deep Gaussian process is minimised when it is as close to its prior as possible. For classification, this is achieved if all members of each class are mapped to a single point in feature space, meaning that only a single support vector is necessary to fit each class, allowing the model to revert to its prior everywhere else. This encourages the model to do as much work as possible using the feature extractor rather than the 'well behaved' feature space model.

6.4 Deterministic Uncertainty Quantification

Above, we have described how to implement a bilipschitz regularisation scheme. We now give an overview of how to implement the 'distance sensitive' final layer classifier. While in principle, any distance sensitive output classification can be used, such as centroid class prototypes as in our paper Amersfoort et al. (2020), or a Gaussian mixture model as in Mukhoti et al. (2021), in this thesis we focus on the Gaussian process case, as the real contribution outlined in this section is related to the regularisation scheme and not the specific method used in the final layer. Gaussian processes, though, are in many ways the nicest from a theoretical perspective, and are the method we use in the concrete implementation presented in the following chapter.

6.5 Approximate Gaussian Processes

A Gaussian process is a random process, that is, a collection of random variables $\{y_x\}$ indexed by $x \in X$ such that for every finite collection of indices $\{x_i \mid x_i \in X, i \in [1..k]\}$ the joint distribution of the vector $Y_{x_1,x_2,\ldots x_k} = (y_{x_1}, y_{x_2}, \ldots y_{x_k})$ is a multivariate Gaussian. As a multivariate Gaussian can be specified by its mean and covariance, a Gaussian process can always be described by a mean function $\mu : X \to \mathbb{R}$ and a covariance function $k : X \times X \to \mathbb{R}$ which define the mean and covariance of the joint Gaussian between any two index points. The joint multivariate Gaussian over a set of index points $x_1, x_2, x_3 \ldots$ has mean vector

$\Big[\mu_(x_1), \mu(x_2), \mu(x_3)\dots\Big]$ and a covariance matrix $\Sigma[i,j] = k(x_i, x_j)$ The covariance function is frequently called the *kernel* of the Gaussian process.

In terms of machine learning, the mean function is typically assumed to be zero, and the covariance function is the main object of interest [6] The covariance function specifies the model. There is a great deal of flexibility in the specification of the covariance function, with the main restriction being that it is *positive semi-definite*, a constraint that arises naturally from the fact that it must give a valid positive semi-definite covariance matrix for any vector of indices.

In a machine learning context, the index set represents the input points and the random variable y the unknown output. Given a training set X, Y and a test point x^* with unknown output and a Gaussian likelihood over the observations of y with standard deviation σ_n^2, a Gaussian process with a given covariance function k has a Gaussian joint distribution over the corresponding output y^*

$$\begin{bmatrix} Y \\ y^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} k(X,X) + \sigma_n^2 I & k(X,x^*) \\ k(x^*,X) & k(x^*,x^*) \end{bmatrix}\right) \tag{6.4}$$

The marginal distribution over the test output, easily obtained via the marginalisation of Gaussians, is $y* \mid X, Y \sim \mathcal{N}(\mu^*, \Sigma^*)$, where the mean and variance are (Rasmussen and Williams, 2006)

$$\mu^* = k(x^*, X)\Big[k(X,X) + \sigma_n^2 I\Big]^{-1} Y \tag{6.5}$$

$$\Sigma^* = k(x^*, x^*) - k(x^*, x)\Big[k(x,x) + \sigma_n^2 I\Big]^{-1} k(X, x^*) \tag{6.6}$$

However, while exact inference in Gaussian processes is simple in the case of regression, this analytic solution has disadvantages. Firstly, it requires the assumption of a normal likelihood over the target variable y, which may not always be appropriate. More significantly, it requires inverting the matrix $k(X,X)$, which

[6] This is because the chief attraction of the Gaussian process is the ability to perform closed form Bayesian inference or approximate inference with reasonable bounds. If the mean function is learned, then it has to be learned in a non-Bayesian way, defeating the main attraction of the model.

is $n \times n$ for n training points, making the complexity of finding the analytic solution cubic in the size of the dataset.

As a result, a major focus of GP research has been designing approximate inference schemes. We describe two here, which are the most useful in the context of deep kernel learning.

Random Fourier Features

The Moore-Aronszajn theorem (Aronszajn, 1950) states that every kernel k has a unique corresponding Hilbert space of functions ϕ, and vice-versa. So rather than thinking about kernels directly, we can consider GP's as being based on the inner product of feature functions ϕ. As outlined in Rasmussen and Williams (2006), chapter 2, while it is arguably more elegant to define GPs directly from the kernel function, they can be equivalently defined by considering Bayesian linear regression with parameterised features ϕ, and then using the kernel trick to re-write this purely in terms of the inner products between feature vectors $\phi(x)$.

This allows replacing inner products between feature vectors with kernel functions which can correspond to infinite dimensional feature spaces provided we can compute inner products between these feature vectors, which allow GPs to have their elegant non-parametric properties. If the implicit feature space is indeed infinite dimensional, then the feature vectors are of more theoretical than practical interest as it is clearly impossible to directly work with them computationally. However, as mentioned, using the kernel trick makes inference scale poorly in terms of the number of input datapoints, wheras linear models on the feature space can scale with either the feature dimension or the number of datapoints, whichever is smaller. To avoid this complexity with large datasets, one possibility is to consider approximating the kernel with a finite dimensional feature space which maintains as many of its properties as possible.

A commonly used class of kernels are *shift invariant*, that is, those which only depend on the *distance* between their input points, that is $k(x, y) = k(x - y) = k(\tau)$. This is particularly relevant for this chapter, for a shift invariant kernel is ‘distance sensitive' in exactly the way that we want - the covariance between two datapoints depends only on the distance between them, regardless of their position. In this case, Bochner's theorem (Rasmussen and Williams, 2006) guarantees that a positive definite kernel $k(\tau)$ is the Fourier transform of a positive measure μ, so we can write $k(x-y) = \int_{\mathbb{R}^D} e^{2\pi i s\cdot(x-y)} d\mu(s)$. Note that, equivalently, defining $\phi_s(x) = e^{2\pi s\cdot x}$, this is the same as taking the expectation of the inner product of feature vectors ϕ under the distribution $\mu(s)$, that is $k(x-y) = \mathbb{E}_\mu\left[\phi_s(x)\phi_s^*(y)\right]$.[7] We can therefore get an unbiased estimate of the kernel by sampling m frequencies s from $\mu(s)$, then using the *finite* feature space $z(x) = [\phi_{s_1}(x), \phi_{s_2}(x), ...\phi_{s_m}(x)]$.

This can be extremely convenient, especially for the Gaussian kernel $k(x,y) \propto e^{\lambda(x-y)^2}$: since the Fourier transform of a Gaussian is another Gaussian, the frequencies s can simply be drawn from a Gaussian distribution.

This scheme was first proposed in Rahimi and Recht (2008). It is particularly convenient for using with deep learning as it approximates a Gaussian process in a *parametric* way, which means that it fits more easily into the standard deep learning workflow than non-parametric approximations; a Bayesian linear model on these finite randomised features is a form of approximate GP.

However, while there are strong theoretical guarantees on the error introduced by this approximation (Rahimi and Recht, 2008), it is not perfect. In particular, while it can be considered an approximate GP, it does not really directly approximate the GP with the corresponding kernel function: rather it changes the kernel to make inference easier to perform. The behaviour of this model is very different to the original GP, especially in the limit: as it is parametric, the posterior over the weights of such a model will saturate to a delta function in the limit of data. This means that the model will eventually extrapolate confidently with negligible epistemic uncertainty, in contrast to a truly non-parametric GP which reverts to the prior even in the limit of infinite data.

[7] This is an example of what it means for a feature vector to be 'infinite dimensional': the feature vector ϕ is indexed by s, which is a real number.

Inducing Point Variational Inference

As mentioned, the downside of the random kernel expansion mentioned in the previous section is that it is not really performing approximate inference in a GP model: rather, it replaces the GP model with a different model designed to approximate the GP, in which inference is easier. It would be desirable to maintain the non-parametric property of GP inference. However, as mentioned, exact inference is prohibitively expensive, as equation 6.6 scales as $\mathcal{O}(N^3)$ due to the inversion of the matrix $K_{XX} := k(X, X) + \sigma_n^2 I$.

Inducing point methods address this scaling by introducing a set of m inducing points $\{z_i, f(z_i)\}$. These function as pseudo-datapoints: rather than making inference directly from the dataset X, y, we will approximate the GP posterior using exact inference conditioned on the inducing datapoints, with z_i functioning as artificial training points and $f(z_i)$ as artificial targets. The locations z_i and values $f(z_i)$ of these inducing points can be chosen to minimise an ELBO on the training set between the approximate posterior q parameterised by GP inference with the inducing points, and the true posterior (Titsias, 2009). By using a non-Gaussian likelihood, inducing point methods can be adapted for classification Hensman et al. (2015).

Variational methods can be trickier to implement than parametric approaches like the RFF, but as mentioned, they maintain some desirable properties. In particular, since the posterior is defined as a Gaussian process posterior, it has similar behaviour, maintaining the property of the target stochastic process that the model reverts to the prior away from the inducing points, no matter how much data the model is exposed to. This is a very important property for our purposes, as one of our key motivations for exploring this kind of model is controlling the behaviour of the model out of distribution.

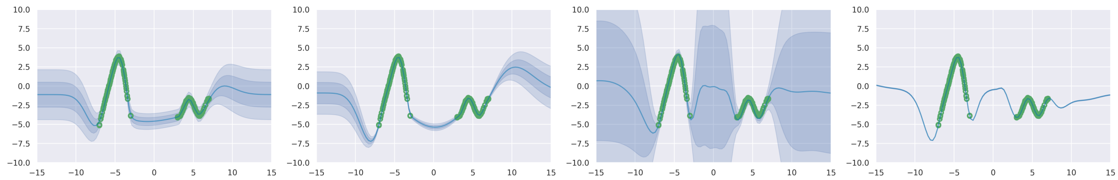
7
Some Empirical Results

Contents

7.1 Introduction

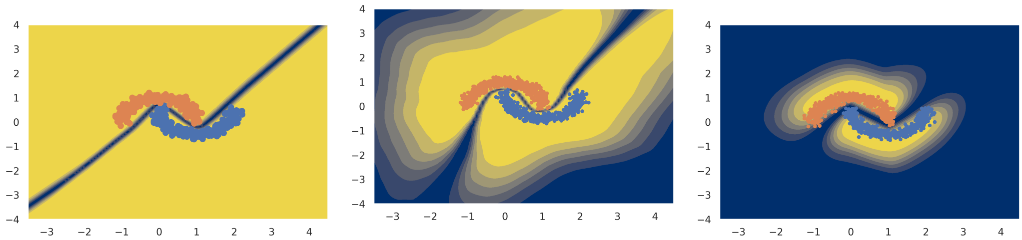
Authorship Statement

This chapter is substantially based on the draft paper “Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression”, Amersfoort, **Smith**, Jesson, Key, et al., 2021, *arXiv preprint arXiv:2102.11409*, on which I was second author. Most of the experimental results in this paper were mainly implemented by the first author Joost van Amersfoort, and the causal inference experiments by Andrew Jesson. My main contributions to this paper were in writing and conception, the study and implementation and of the Lipschitz constant in Batch-Norm, and the implementation of the RFF approximation in our library in order to study its effects in Section 7.2.1



(a) DUE - 1K **(b)** DUE - 1M **(c)** SNGP - 1K **(d)** SNGP - 1M

Figure 7.1: 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) datapoints 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.



(a) ResNet + Softmax **(b)** GPDNN (Bradshaw et al., 2017) **(c)** DUE (us)

Figure 7.2: We show uncertainty results on the two moons dataset. Yellow indicates high confidence, while blue indicates uncertainty. In Figure 7.2a, a simple feed-forward ResNet with a softmax output is certain everywhere except on the decision boundary. In Figure 7.2b, we see that GPDNN, which uses a simple Feed-Forward Network as feature extractor, is certain even far away from the training data. In Figure 7.2c, we show DUE, which has the appropriate restrictions on the feature extractor (residual connections and spectral normalization) and obtains close to ideal uncertainty on this dataset.

> However, I felt that it was important to include these experimental results in a unified and coherent presentation of the 'Bilipschitz' model family, as without the context of these models having good experimental performance the more theoretical and conceptual work makes little sense.

In the previous chapter, we gave an overview of the key ingredients - spectral normalisation and approximate Gaussian processes - necessary to implement a 'bilipschitz' model. In this chapter, we describe a concrete implementation, and present some empirical results using this type of model, demonstrating their promise. We refer to the model in question as 'DUE' (Deterministic uncertainty estimation), as one advantage of this kind of kernel method is that it requires only a single pass through the feature extractor, unlike ensemble methods or Bayesian averaging over weights which require multiple forward passes for a single evaluation.

Algorithm 7.1 Algorithm for training DUE

Definitions:

- Residual NN $f_\theta : x \to \mathbb{R}^J$ with feature space dimensionality J and parameters θ.
- 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)
- Learning rate η, loss function $\mathcal{L}$

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

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.

Initial length scale:

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

for minibatch $\boldsymbol{x}_b, \boldsymbol{y}_b \subset X, Y$ **do**

  $\theta' \leftarrow$ spectral_normalization(θ)

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

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

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

end for

7.1.1 The DUE Model

As discussed in the preceding chapter, **DUE** can be seen as a form of deep kernel learning, such as GPDNN (Bradshaw et al., 2017). We use a residual network, with spectral normalisation applied to the residual connection, as our feature extractor, as described in the previous chapter. Specifically, we use a feature extractor based on the Wide ResNet of Zagoruyko and Komodakis (2016), adding spectral normalisation but otherwise leaving the architecture unchanged. We make several changes to the training process used by Bradshaw et al. (2017) for their deep kernel model. We use the feature vector of last convolutional layer of a large model, which is 640 dimensional in the case of the WRN, directly as the GP input rather than using an additional downsampling layer. We find that, with our feature extractor, no pre-training is necessary and training is stable with a single set of hyper-parameters for both the variational and model parameters. The inference procedure is described in detail in Appendix B.1. The training procedure is described in Algorithm 7.1.

As discussed in Section 6.2.1, we use power iteration to enforce a spectral normalisation on our network.

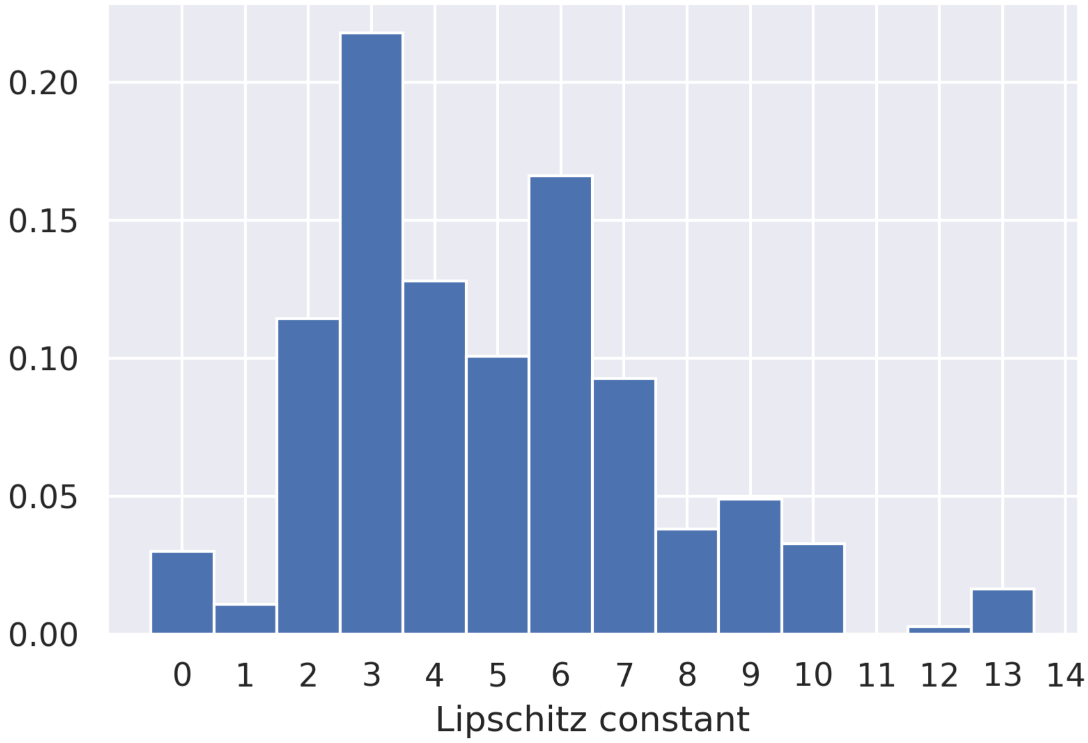


Figure 7.3: A density of the Lipschitz values in batch normalization layers, averaged across 15 WRN models that were trained with Softmax output and without spectral normalization (exactly following Zagoruyko and Komodakis (2016)). We see that many of the constants are significantly above 1, highlighting that batch normalization has significant impact on the Lipschitz constant of the network.

In addition, we extend spectral normalization to batch normalization by dividing the weight γ of the batch normalization by the (scaled) Lipschitz constant, as discussed in Section 6.2.1. We note that the related work of Liu et al. (2020) use standard BatchNorm without accounting for the Lipschitz constant, but we find that batch normalization layers in trained ResNets have a relatively high Lipschitz constant, up to around 12, with 95% of the channel-wise Lipschitz constants being greater than one, as shown in Figure 7.3. We ensure the entire network's upper Lipschitz constant is bounded by using this spectrally normalised variant of BatchNorm.

	bilipschitz		non bilipschitz	
	AUROC	Accuracy	AUROC	Accuracy
SV-DKL[a]	**0.959±.001**	95.7±0.06	0.498±.001	93.6±0.05
GPDNN[b]	**0.958±.005**	95.6±0.04	0.876±.004	93.7±0.10

Table 7.1: AUROC on CIFAR10 vs SVHN for two DKL methods – SV-DKL trained with feature extractors with and without a bilipschitz constraint. Non bilipschitz uses VGG-19 (Simonyan and Zisserman, 2014); bilipschitz uses a WRN with spectral normalization. Both models obtain high accuracy, matching standard softmax models. SV-DKL without bilipschitz obtains poor uncertainty: distinguishing in- and out-of-distribution data is no better than chance. GPDNN without bilipschitz obtains better but still poor uncertainty. The cell highlighted in grey – GPDNN with our architectural changes to the feature extractor – is DUE.

[a] Wilson et al., 2016a.
[b] Bradshaw et al., 2017.

7.2 Related Work

The single forward pass uncertainty methods most similar to DUE are DUQ (Amersfoort, **Smith**, Teh, and Gal, 2020)[1] and SNGP (Liu et al., 2020), as discussed in the introduction and in the context of enforcing the bilipschitz constraint. DUQ classifies points based on their proximity to centroids in feature space, and so, while effective, is not readily applied to regression problems, while DUE can be applied to either classification or regression with no difficulty. In addition, the gradient penalty used by DUQ is difficult to optimise and slow to compute, requiring an additional backward pass to compute the second order gradient. 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 and Recht, 2008) in combination with a Laplace approximation for the non-conjugate Softmax likelihood (Rasmussen and Williams, 2006).

[1] Though I was second author on this paper, the empirical results presented in it are not discussed in this thesis. The main reason is that, while DUQ contained important ideas in terms of proposing the essential idea of bilipschitz-type regularisation, in practice subsequent methods are a both an empirical improvement on DUQ and both cleaner conceptually and in implementation, so in the interests of presenting the clearest possible presentation of the state of the art as I currently see it, I decided that the empirical results of DUQ were not particularly useful, especially as my contribution to the project was more on the theoretical/conceptual side, and the specific implementation tricks used to get DUQ to work well, while ingenious, should be credited to Joost van Amersfoort. The key features of the DUQ model are outlined in Section 6.2.2

Several methods which also require only a single forward pass through a network, but which are not based on kernel learning, have been proposed. For instance, Prior networks (Malinin and Gales, 2018) follow a similar approach of changing the likelihood from a softmax to a Dirichlet based likelihood. However, prior networks require out of distribution examples at training time. Similarly, function-space variational inference (Sun et al., 2019), requires sampling from out of distribution points at training time. This requires the user of the method to define a way to sample out of distribution points. It is not obvious how to do this in many applications, and the behaviour of such a method in practice is likely to depend strongly on this choice. In contrast, our method doesn't require out of distribution training data, and so avoids this problem.

7.2.1 Inducing Points versus RFF Approximation

GP approximations were discussed in Section 6.5. Recall that inducing point GP approximations maintain the non-parametric properties of the full GP, by calculating the kernel between the input and a set of learned 'inducing points', while the RFF approximation sacrifices this. With the RFF approximation, for any *finite* number of features, the kernel is approximated as a linear model on these features. Because of this, the RFF GP's epistemic uncertainty will concentrate to zero everywhere as the number of training points increases, even in areas where there is no training data. Various extensions of RFF (and its close relative, the sparse spectrum GP approximation) have attempted to fix this problem, see for example Gal et al. (2015).

In Figure 7.1, 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[2]: it is wide in areas where there is no training data at 1K datapoints, but very narrow in the same regions at 1M datapoints. This is because the approximation is parametric, the support of the posterior concentrates in the limit of data. In contrast, as the variational posterior is a non-parametric method, even in the limit

[2] In fact, in a linear model, the variance of the posterior over weights can *only* decrease with more data

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

Table 7.2: Results on the CIFAR-10 dataset, and distinguishing between CIFAR-10 and SVHN by uncertainty. All results use a WRN as feature extractor and are the average and standard error of 5 runs. 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.

[a]Lakshminarayanan et al., 2017.
[b]Zagoruyko and Komodakis, 2016.
[c]Amersfoort et al., 2020.
[d]Liu et al., 2020.
[e]Wilson et al., 2016a.

of data it reverts to the prior away from the support of the training data (or more accurately, away from the inducing points).

SNGP uncertainty was estimated using the exact method with a ridge penalty of 1. All other hyper-parameters were shared between the two methods, and are listed in Appendix B.2.

7.3 Experiments

7.3.1 Feature Collapse in DKL

We analyse the problem of feature collapse in DKL in Figure 7.2 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)). Further details are provided in Appendix B.2. 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 7.2c) quantifies uncertainty in a close to ideal way for the two moons dataset: certain on the training data, uncertain away from it and in-between the two moons. Figure 7.2b highlights the importance of our contribution; the uncertainty estimation of a standard Feed-Forward Network in combination with DKL suffers from feature collapse and is certain away from the training data, demonstrating the importance of the neural architecture to getting deep kernel learning to behave in the way that we want.

7.3.2 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, Hinton, et al., 2009). We follow Zagoruyko and Komodakis (2016) and use a 28 layer model with BasicBlocks and dropout. Importantly, we can use their hyper-parameters (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.

We perform an ablation over more inducing points in Table 7.3, showing that adding more inducing points, while it does not harm performance, has only a small benefit over using a small number in this model. This is likely because the feature extractor is able to strongly cluster the classes in feature space, as shown in Figure 7.4. This is consistent with the observation that the ELBO of the feature space GP encourages the model to allow the GP to be as close as possible to its prior.

7.3.3 Uncertainty out of distribution

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) using each model's uncertainty metric. Note that this was found to be a particularly

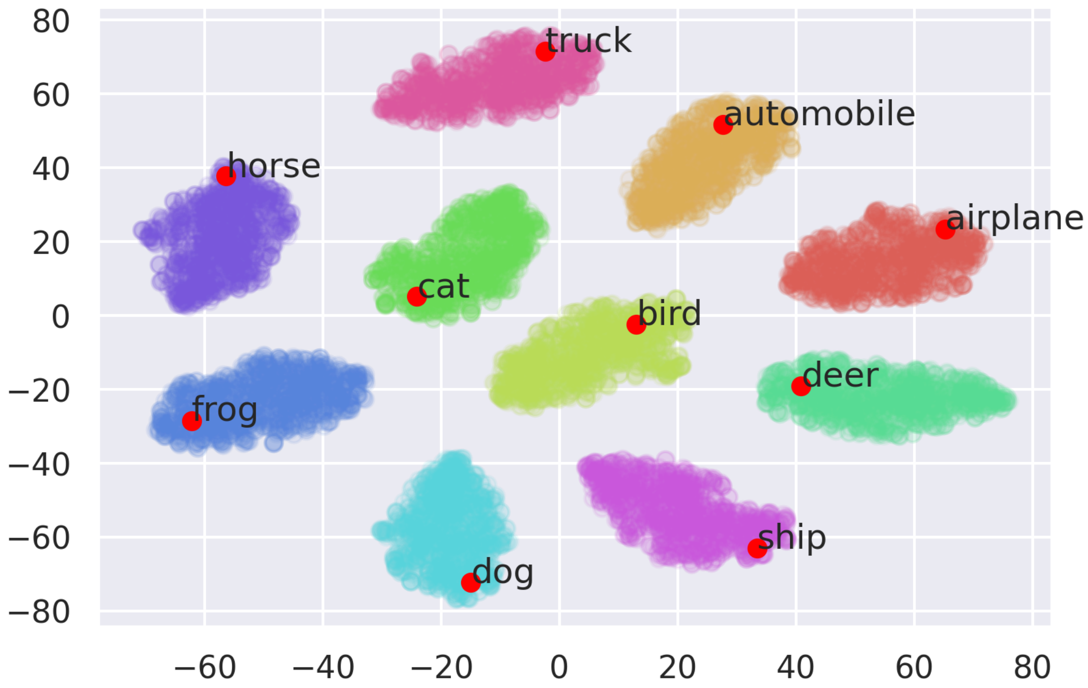


Figure 7.4: Visualisation of the learned feature representations of the training set of CIFAR-10. We use T-SNE (Van der Maaten and Hinton, 2008) to reduce the dimensionality down to 2D and we colour the points by their class label. We overlay (in red) the inducing points location, which we label by computing 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 normalization.

difficult dataset pair in Nalisnick et al. (2019b). In Table 7.1, 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 hyper parameters (including mini-batch-size) as DUE. This demonstrates the need for a regularisation scheme for obtaining good out-of-distribution uncertainty with deep kernel learning; all these methods achieve reasonable test accuracy, but the uncertainty cannot be used to distinguish in and out of distribution data without our regularisation scheme. This highlights the importance of our contribution: additional constraints on the feature extractor are crucial for good uncertainty performance in DKL.

Table 7.2 compares DUE to several baselines. We train DUE using 10 inducing points and spectral normalization constant 3, set in similar fashion to Liu et al. (2020) by choosing the lowest value that does not significantly affect accuracy. All methods

Table 7.3: 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 a large number of 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

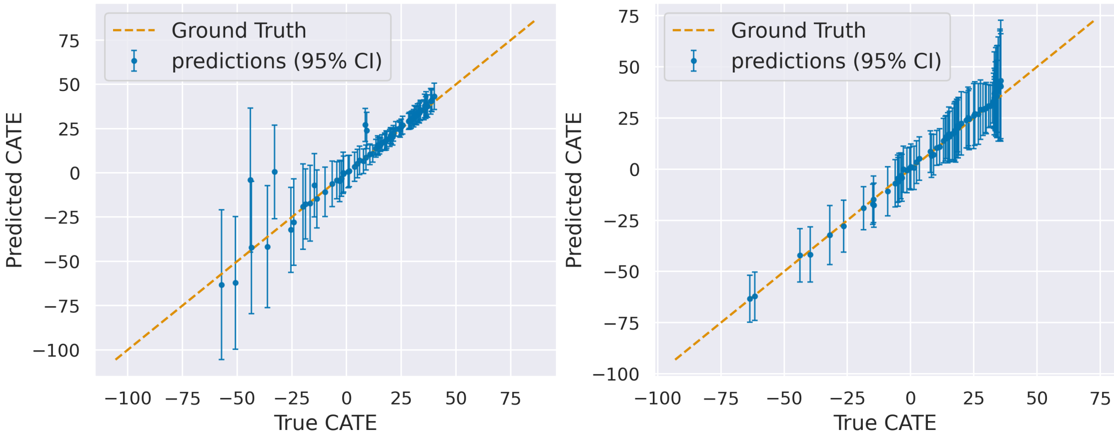
Table 7.4: ECE calibration results on CIFAR-10 with 15 bins and no post-hoc scaling. DUQ and SNGP results obtained from Liu et al. (2020).

Method	ECE
DUE	0.01795 +/- 0.0015
DUQ	0.034 +/- 0.002
SNGP	0.018 +/- 0.001

compute uncertainty using the predictive entropy, except DUQ which uses the closest kernel distance (Amersfoort et al., 2020). We see that DUE and SV-DKL, with DUE feature extractor, outperform all competing single forward pass uncertainty methods. DUE is competitive with Deep Ensembles in terms of uncertainty, but only requires a single forward pass and is therefore five times faster to train and evaluate. SV-DKL with the DUE feature extractor has similar performance 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 variational approximation also leads to practical improvements over the RFF, as was argued theoretically in Section 7.2.1, as both outperform SNGP in terms of uncertainty estimation.

7.3.4 Uncertainty in regression for personalised healthcare

To demonstrate the performance of DUE on regression tasks, we focus on a new benchmark on personalised healthcare (Jesson et al., 2020). The task is to predict the responses of individuals to a particular treatment. This individualised prediction



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

Figure 7.5: 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.

is only possible if there is sufficient data available to assess how they might respond. In other words, 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. This is called the overlap assumption. 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 accurately distinguish between patients where the outcome is well represented in the training data and patients where the model is extrapolating, otherwise an individual could receive treatment even if their response to treatment is not known, which can result in undue stress, financial burdens, or worse. The overlap assumption is essentially a test for data which is in distribution, and so this benchmark is clearly an application where controlling the behaviour of the predictor out of distribution should improve performance, while still requiring good predictive performance in order for the model to be useful, in contrast to standard benchmarks such as UCI (Dua and Graff, 2017), which only require good test log likelihood.

We quantify the personalised effectiveness of a treatment using the Conditional Average Treatment Effect (CATE) (Abrevaya et al., 2015), computed as the

Method	IHDP Cov. (50% def.) *random*	*uncertainty*	IHDP (10% def.) *random*	*uncertainty*
BART[a]	2.6±.2	1.8±.2	1.90±.20	1.60±.10
BTARNet[b]	2.2±.3	1.2±.1	1.10±.03	0.76±.03
BCEVAE[c]	2.5±.2	1.7±.1	1.80±.06	1.47±.08
DKLITE[d]	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**

Table 7.5: Comparing the performance in terms of RMSE of several treatment effect and uncertainty estimation models under "random" and "uncertainty" based referral to an expert. The first three rows were obtained from Jesson et al. (2020), DKLITE was evaluated using the author's open source implementation and the ensemble of TARNet was reimplemented. DUE outperforms all alternative methods, while being 5x faster to train and evaluate than the ensemble.

[a]Chipman et al., 2010.
[b]Shalit et al., 2017.
[c]Louizos et al., 2017.
[d]Zhang et al., 2020.

difference between the expected effect of treatment and no treatment. A patient is represented by a set of features x, their health outcome by y, and whether they received the treatment or a control is represented by the binary variable t. Given a model for the distribution of $p(y \mid x, t)$, the CATE is simply the expected difference of the conditional health outcome;

$$\text{CATE}(x) = \mathbb{E}\left[y_1 - y_0\right], \qquad y_1 \sim p(y \mid x, t=1), y_0 \sim p(y \mid x, t=0) \tag{7.1}$$

and similarly, we can quantify the uncertainty in the CATE by calculating the variance of $y_1 - y_0$.

Obtaining a model for $p(y \mid x, t)$ from observational data where we have patient features x, the applied treatment t and the observed outcome y is simply a standard regression task. Using DUE, 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 feature extractor. The Monte Carlo sampling is over the feature space GP, which is relatively cheap). 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 datasets 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, a given proportion of 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 7.5, we compare DUE with Bayesian TARNet, which is the standard TARNet (Shalit et al., 2017) extended with MC dropout (Gal and Ghahramani, 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 7.5 summarises our results compared to several baselines (detailed in Appendix B.2.3) and shows that DUE has improved performance and uncertainty estimates better suited to rejection policies than other uncertainty-aware methods.

7.4 Conclusion and Limitations

The results presented in this chapter demonstrate that DUE outperforms alternative single forward pass methods on CIFAR-10, and obtains SotA performance in a personalised medicine regression benchmark, outperforming specially designed methods. These results show that DUE mitigates problems with uncertainty in DKL observed in prior literature, and makes DKL a viable method for uncertainty estimation and a practical tool for improving reliable AI, as well as a potential direction for future work to build on. DUE offers a combination of practical speed, comparable ease of use to standard neural networks, and strong performance which may also be useful for practitioners.

However, there are important caveats to this empirical success. While these results demonstrate that DUE is empirically competitive with other solutions, like deep ensembles, on the measures of uncertainty we have evaluated, there remain important avenues we have not explored, including robustness under an adversarial threat model. As we outline in the next chapter, there are reasons to be pessimistic about this; while the regularisation scheme used to achieve this empirical progress was motivated by a connection between bilipschitz mappings and avoiding feature collapse, this theory does not apply to the models actually used in practice as easily as previous work on the subject has assumed.

8

Understanding the Bilipschitz Constraint

Contents

8.1 Introduction

Authorship Statement

This chapter is based on the draft “Can convolutional ResNets approximately preserve input distances? A frequency analysis perspective”, **Smith**, Amersfoort, Huang, Roberts, et al., 2021, *arXiv preprint arXiv:2106.02469*, currently under review for ICML. It is substantially my own work. Haiwen Huang ran and organised results from many of the experimental sweeps in Section 8.6, though writing the software library to do so was a collaborative process. Discussions between the co-authors were invaluable in shaping the presentation of the theoretical argument.

As we have seen in the previous chapter, the 'bilipschitz' architecture of a spectral constraint and residual connections is effective empirically at improving the performance of deep kernel learning. We have also discussed the motivation for this kind of constraint, stemming from the idea of preventing feature collapse, and enforcing a bilipschitz constraint, or approximate distance preservation, on the feature extractor.

However, while this explanation motivated the work described above, it is not entirely satisfactory. While these regularisation schemes were motivated as imposing a bilipschitz constraint on the model, it is in fact *mathematically impossible* for mappings which do not preserve the dimensionality of the input space to be bilipschitz, as we demonstrate in detail below. A necessary implication of this argument is that it is impossible to prevent feature collapse entirely, if the model is not dimensionality preserving. For the model of the previous chapter, which used a deep model which mapped CIFAR 10 (naively $3 \times 32 \times 32 = 3072$ dimensional) into a 640 dimensional feature vector, there must exist at least some distinct vectors in the input space which are mapped to identical outputs. As such, the idea that the bilipschitz regularisation scheme works because it prevents feature collapse, while true in models which preserve dimensionality such as those used in Behrmann et al. (2019), clearly cannot explain the effectiveness of the regularisation scheme in the models discussed in the preceding chapter.

This observation raises the obvious question of why these regularisation schemes are effective, if it isn't because they prevent feature collapse. In this chapter, we introduce an alternative explanation; we demonstrate that, under mild assumptions, the regularisation scheme used in the previous chapter can enforce preservation of distances between *low frequency projections* of the input images. This is based on the observation that the dimension reduction layers in convolution networks are generally of a particular kind; strided convolutions or other spatial pooling layers. As we shall see, the effect of these on the input signal can be characterised in the frequency domain, offering a means of re-establishing some guarantees under particular assumptions about the input. The prevention of feature collapse between the low-frequency projections of inputs is relevant because natural images are typically concentrated in the low frequencies.

The contributions of this chapter are as follows

1. We show that models including downsampling layers cannot be bilipschitz.
2. We prove, under mild assumptions, that we can establish a lower bound on the low-frequency distance between the feature maps of residual blocks given the low-frequency distance between their inputs. Since the low-frequency components of images will pass through a downsampling operation unchanged, this establishes a lower bound on the feature collapse even after downsampling when our assumptions hold.
3. We describe a simple constructive algorithm for finding image pairs which exhibit feature collapse, that is, where $||x - y||$ is large but $f(x) \simeq f(y)$.
4. We verify these theoretical claims empirically, investigate the behaviour of the model in terms of feature collapse when our assumptions are violated, and discuss the implications for future model design.

8.2 The "Bilipschitz" Constraint

Firstly, we review the original justification for why the spectral normalisation scheme should preserve distances approximately, as presented in prior work (Amersfoort et al., 2021; Liu et al., 2020) and the previous chapter. This applies also to gradient penalties, as mentioned in Section 6.2.2. We then prove that models which are not dimensionality preserving cannot be bilipschitz. This shows that existing motivations for this regularisation scheme based on the idea that it enforces bilipschitzness are flawed. In later sections, we provide an alternative explanation for this empirical success. For brevity, intuitive proof sketches are provided in the main body, with full proofs in Appendix C.2 where applicable.

8.2.1 Background: spectral normalisation

The spectral normalisation scheme used in Liu et al. (2020) and the previous chapter, was first introduced by Behrmann et al. (2019) in the context of designing invertible models:

Theorem 8.1 (Behrmann et al., 2019)**.** *Let $f(x) = x + g(x)$ be a residual block, and let $\mathrm{Lip}(g) = L$ with $L < 1$. Let $||x||$ denote the Euclidean norm of the vector x. f is bilipschitz, with*

$$\frac{1}{1+L}||x-y|| \leq ||f(x) - f(y)|| \leq (1+L)||x-y||.$$

This result suggests regularising the Lipschitz constant of a neural layer g using spectral regularisation, as discussed in Chapter 6. In this way, we can constrain all residual blocks in our network to satisfy the assumption of this theorem. Though, note that the theorem only enforces Bilipschitzness if the spectral normalisation constraint $\mathrm{Lip}(g)$ is below 1.

This theorem, however, only applies to residual connections where x and $g(x)$ have the *same* dimensionality (otherwise the meaning of the expression $x + g(x)$ is undefined). However, the ResNet architectures used in prior work relying on this spectral constraint include strided convolution layers in g, reducing the dimensionality, and typically apply a strided 1x1 convolution to the 'skip' connection. The ResNet is therefore performing an operation $Ax + g(x)$, where A is a downsampling operation. Prior work using this penalty (Amersfoort et al., 2021; Liu et al., 2020; Mukhoti et al., 2021), including the results presented in the previous chapter, has implicitly assumed in justifying the use of this scheme that this does not affect the bilipschitzness of the resultant model, even though Theorem 8.1 does not straightforwardly apply in this case.

8.2.2 Non dimension preserving maps are not bilipschitz

In fact, not only does Theorem 8.1 not apply, but it is impossible for a model of this form to be bilipschitz:

Theorem 8.2. *Let f be a function from $\mathbb{R}^n$ to $\mathbb{R}^m$, with $m < n$. Then f is not bilipschitz.*

Proof. This follows from the fact that bilipschitz mappings are homeomorphisms and that $\mathbb{R}^n$ and $\mathbb{R}^m$ are not homeomorphic unless $m = n$ (Brouwer, 1911; Druţu and Kapovich, 2018). A full proof is provided in Appendix C.2. □

This theorem is a general statement which is independent of the precise function or architecture choice – this theorem applies to any mapping between high and low dimensional spaces. We don't claim this is a particularly novel result in purely mathematical terms – it follows fairly directly from well established mathematical facts about bilipschitz functions and topology. However, it appears to have been overlooked on prior work using bilipschitzness in neural networks. This theorem makes it clear that the properties of any architecture which performs downsampling overall *cannot* be explained by this architecture being bilipschitz, as claimed in Amersfoort et al. (2021) and Liu et al. (2020) because bilipschitzness is *mathematically impossible* to achieve for this kind of model. This motivates our investigation into weaker properties, analogous to bilipschitzness, which might explain the effectiveness of this regularisation scheme without contradicting this theorem.

8.3 Frequency Analysis

In this section, we motivate the use of the frequency domain for analysing networks that use convolutions followed by downsampling operations, introducing necessary concepts in Section 8.3.1. We then present the argument behind our main theoretical claims in Section 8.4.

8.3.1 Downsampling and feature collapse in CNNs

Theorem 8.2 straightforwardly applies to models which treat the input as a vector in an unstructured space. However, we are interested in particular in convolutional models, where both the feature maps and downsampling operations have a spatial structure. As observed by Zhang (2019), downsampling operations in convolutional networks can be thought of as a dimensionality preserving dense operation followed by an image decimation step.[1] For example, a strided convolution is identical to a standard convolution followed by an image subsampling step where we drop pixels from the output.

Strided convolutions are our running example because of their centrality in the widely-used ResNet architecture. Note that this also applies to other linear downsampling operations like average pooling, which is a special case of strided convolution.

The ability to rewrite this downsampling operation is important, because it means that mathematically, instead of thinking about strided convolutions as atomic operations, we can always split a strided convolution into a normal convolution and a decimation operation. In fact, this can be done in general for an entire residual block, that is

Lemma 8.1. *Let $f(x) = A(x)+g(x)$ be a downsampling residual block, where A and g are both compositions of dimension preserving convolutions, strided convolutions, pointwise nonlinearities and normalisation layers. Let D be a decimation operation. We can equivalently express f as $f(x) = D(A'(x) + g'(x))$ where A', g' contain no strided convolutions (i.e are dimension preserving).*

Our lemma is obvious for a single strided convolution, but if the strided convolution is in the middle of g, then the argument is a bit more subtle. We provide a full proof of this claim in Appendix C.2.

This result is useful because it allows us to treat blocks containing strided convolution as a combination of a dimension preserving mapping and a *non-learnable* downsampling operation, decimation. We are thus free to study the effect of convolution blocks on a signal and the effect of decimation on a signal separately. Decimation in particular is also useful because it has been studied extensively in the context of signal processing, and sufficient conditions for this downsampling operation to produce *no* feature collapse are known.

[1]Decimation by a factor n is keeping only every n^{th} sample on a regular spaced grid. In numpy (Harris et al., 2020) syntax, `x[..., ::n, ::n]` implements the decimation by a factor n of the 2D signal `x`

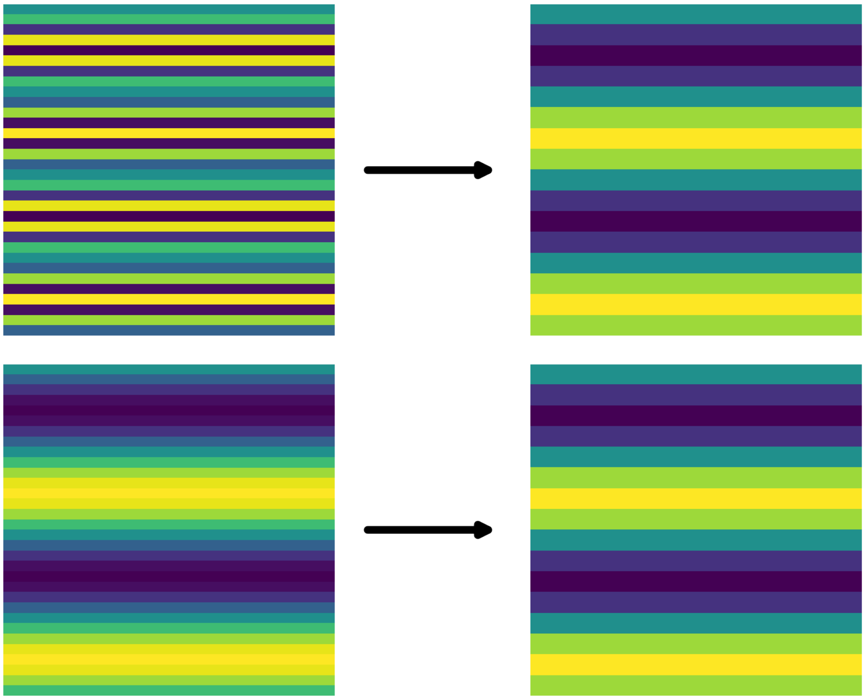


Figure 8.1: Two Fourier modes which are aliases of each other when aliased by a factor 2, with the original images on the left: after decimation (represented by the arrow), the images become indistinguishable. Applying anti-aliasing, as in BlurPool (Zhang, 2019), avoids this problem, but would instead map the top image to zero, as this mode would be removed by pre-decimation filtering.

The Nyquist-Shannon theorem states that, for a given sampling frequency u_s, we can perfectly reconstruct a signal assuming it contains no frequencies above the Nyquist rate $u_s/2$. If there are frequencies above the Nyquist rate, these must be removed before decimation to prevent *aliasing*, where high frequency components of the pre-decimated signal appear as low-frequency components after decimation, as shown in Figure 8.1. Typically, in classical signal processing, a low-pass filter is applied before decimation to reduce these effects, a so-called *anti-aliasing* filter. This does not reduce the amount of feature collapse which occurs, but it makes the behaviour more desirable; anti-aliased downsampling makes signals which differ only in their high frequency content aliases of each other, whereas naive decimation can transform a high frequency signal into a lower frequency alias, as Figure 8.1 shows. The effect of downsampling Fourier modes with an anti-aliasing filter is demonstrated in Figure 8.2 This operation has been proposed as a small modification to standard strided convolutions in Zhang (2019), which improves performance and translation invariance of standard convolution networks. Since it is more correct from a signal processing perspective, generally improves performance, and considerably simplifies the analysis of decimation, from here we always assume that our signal is low-pass filtered just prior to decimation. We discuss some more subtle points about decimation and aliasing in more detail in Appendix C.1.

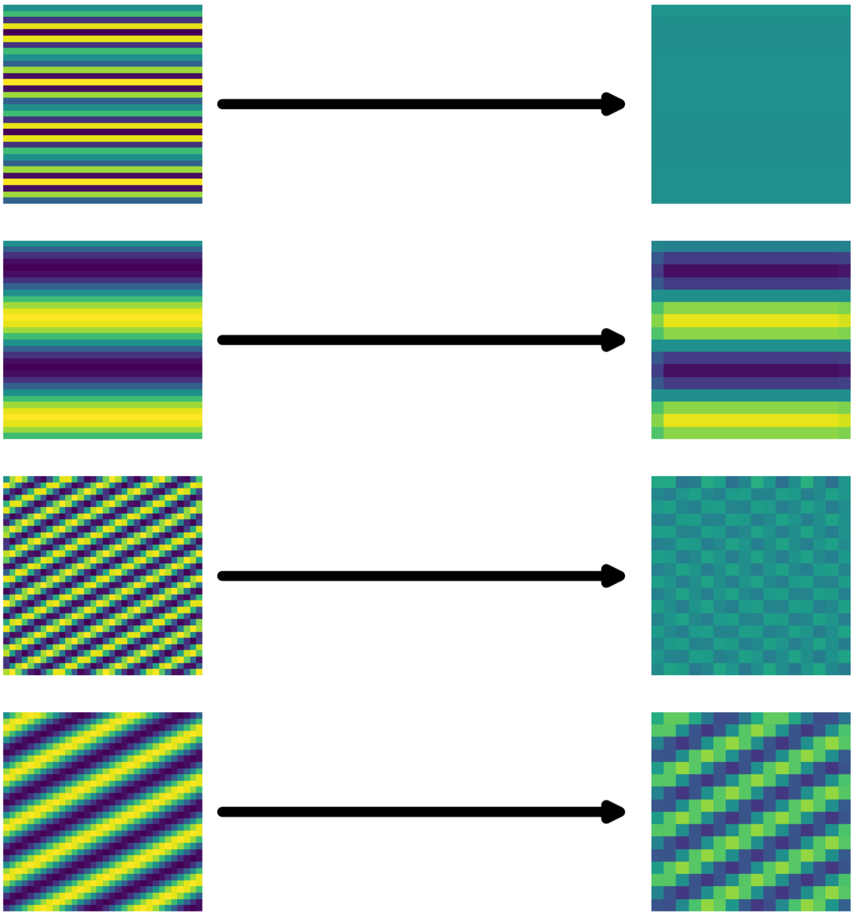


Figure 8.2: Downsampling the Fourier modes shown in Figure 8.1, but applying a low-pass filter before decimating. The two modes are now no longer aliases of each other. On the other hand, we show two additional modes, again one above and one below the Nyquist; the two modes on the first and third row, which are above the Nyquist rate, are now close to aliases of each other. Note in this figure, we use a filter with finite support so that high frequencies are suppressed, but not entirely removed, so content above the Nyquist is present in the output, just with much lower amplitude.

In other words, provided two inputs contain no content above the Nyquist rate, their distances are perfectly preserved (up to isometry) by decimation. If we have two signals (such as feature maps in a neural network) x and y which *are different below the Nyquist rate*, then after decimation they are guaranteed to still be distinguishable, since their low-frequency content will be preserved by this operation. Decimation (with anti-aliasing) only causes complete feature collapse between x and y if they are only different above the Nyquist frequency.

8.3.2 The Discrete Fourier Transform

An essential tool for considering the frequency content of images is the Discrete Fourier transform, or DFT. The DFT takes a signal on a regular n dimensional grid and represents it as a sum of plane waves, functions of the form $\omega(x) = \phi e^{ix \cdot k}$, where k is a wavevector, ϕ a complex number, and i is the imaginary unit. For a signal $z = z_{xy}$ on a 2D $M \times N$ array, indexed by $x = -\frac{M}{2} + 1 : \frac{M}{2}$ and $y = -\frac{N}{2} + 1 : \frac{N}{2}$, its DFT Z_{uv} is an array of the same dimensionality indexed by $u = -\frac{M}{2} + 1 : \frac{M}{2}, v = -\frac{N}{2} + 1 : \frac{N}{2}$. Letting $\omega_N^{nk} = e^{2\pi i \frac{nk}{N}}$, the Fourier transform Z (alternatively $\mathcal{F}(z)$) of input signal z is defined as:

$$Z_{uv} = \frac{1}{\sqrt{MN}} \sum_{x=-\frac{M}{2}+1}^{\frac{M}{2}} \sum_{y=-\frac{N}{2}+1}^{\frac{N}{2}} z_{xy} \omega_M^{-ux} \omega_N^{-vy}. \tag{8.1}$$

The inverse is defined similarly, but with ω_M^{xy} replaced by their complex conjugates, so $z_{xy} = \mathcal{F}^{-1}(Z)_{xy} = \frac{1}{\sqrt{MN}} \sum_u \sum_v Z_{uv} \omega_M^{ux} \omega_N^{vy}$. This formula for the inverse Fourier transform makes clear how the DFT represents an image as a sum of plane waves, with the Fourier coefficient Z_{uv} describing the amplitude and phase with which the corresponding plane wave $f_{xy} = \omega_M^{ux} \omega_N^{vy}$ is present in the image.

The Fourier representation of an image is useful and natural way to think about and manipulate the content of natural images. For instance, classical image compression algorithms such as the JPEG work by using the frequency representation of an image, and quantising high-frequency components to achieve good compression with little perceptual difference (Wallace, 1992), as most of the power of a typical image is located in the low frequencies. They are also useful for our purposes, as as discussed above, decimation (with aliasing) has a particularly simple effect in the frequency domain, as it can be modelled as simply throwing away all the frequency content above a threshold frequency. In addition, the *circular discrete convolution* obeys the convolution theorem: $\mathcal{F}(x * y) = \mathcal{F}(x) \cdot \mathcal{F}(y)$: a convolution in the spatial domain is pointwise multiplication in the frequency domain, and vice versa, so convolutions also have a simple action in the frequency basis. This makes the DFT a natural tool to use to analyse the behaviour of decimation and thus downsampling in convolution networks.

8.4 Frequency Analysis of Residual Networks

In order to use the result that decimation can be studied in the frequency domain, however, we first need to consider the effect of residual networks on their inputs in the frequency domain, which is studied in this section. We are able to show that the action of residual blocks in frequency space can be constrained, then combine this with the results in the previous section to establish some bounds on the distortion of the *low frequency projections* of inputs. The appropriate cutoff for a frequency to qualify as 'low frequency' depends on the architecture of the model, and will be discussed in more detail later in this section.

By 'residual block', we mean a function of the form $f(x) = x + g(x)$, where the residual connection $g = g_1 \circ g_2 \circ ...g_n$, and the components g_i are all convolutions, ReLUs or batch normalisation layers. Recall that, as per Lemma 8.1, we can re-write residual layers which contain downsampling into this form by 'pulling out' the downsampling.[2] H_u is an (ideal) low-pass filter, removing all frequencies in a signal above the cutoff frequency u. $x \cdot y$ denotes pointwise multiplication, and $x * y$ circular convolution. Recall that the power spectrum of an image x is the absolute value of its (complex) Fourier transform, $|X|$.

[2]This implies an additional assumption, which is that only average pooling is applied to the 'identity connection' on x. In practice, in standard architectures, generally speaking a 1x1 kernel convolution is applied here because the number of channels is normally increased while downsampling the spatial dimensions of the feature maps. So the model here is a slightly simplified version of what is typically used in standard architectures, but we think that it is a reasonable theoretical model, and we validate this assumption in the experimental sections. Intuitively, using a linear 1x1 convolution on the channels must duplicate the information in the channels, making feature collapse less likely, but it is more difficult to be mathematically precise about this.

This discussion considers only a single channel. However, this is acceptable because downsampling and nonlinearities also act independently on the channels, and so we can consider the channels independently. Convolutions may mix channels, but, as explained below, they cannot mix frequencies, and we can bound the Lipschitz constant of multi-channel images effectively.

In the following discussion, we treat a single residual block in isolation, before discussing how this can be extended to a sequence of blocks. We often use the term 'image' to refer to the input to a block: this is to emphasise that we are treating the input to the block as a *spatial* signal, but here 'image' can refer to either the input image, or an intermediate feature map of the neural network.

We now proceed to characterise the effect of all ResNet components in frequency space, before using this to establish our main result. Most ResNet components are very simple.

BatchNorm multiplies each channel by a scalar, and adds a constant bias. Multiplication by a scalar does not change the relative scale of the frequency representation of an image, and adding a bias only changes the constant term corresponding to the zero frequency. As discussed in the previous chapter, BatchNorm can also be made Lipschitz.

Convolutions have a simple representation in the frequency domain, as a result of the convolution theorem, discussed above. A convolution acts *independently* on every frequency. In particular, this means that if the Lipschitz constant of the convolution is bounded, the action on *every frequency component individually* has to be bounded since the convolution is a diagonal matrix in the Fourier basis.

ReLUs are nonlinear, and so their effect in frequency space is more difficult to constrain.[3]. In fact, a ReLU acts as a *convolution* in the frequency domain. To see this, consider an image x, and let $m = I[x > 0]$ be the binary mask of the locations in the image where the ReLU is active. Then we can write the ReLU as a pointwise multiplication, $\mathrm{ReLU}(x) = x \cdot m(x)$. Because, by the convolution theorem, this means that the action of the ReLU on X will be a convolution of

[3]The proofs in this section can be easily extended to leaky ReLUs, but we only consider standard ReLUs for simplicity.

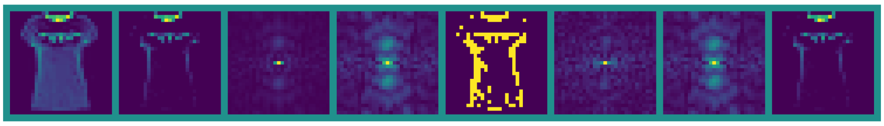


Figure 8.3: An image demonstrating the frequency domain representation of the ReLU, on an image from FashionMNIST. From left to right; a) the original image x, normalised to have zero mean and unit std. dev. so that the ReLU has a non-trivial effect. b) $z = \max(x, 0)$ c) The FT of x, X. d) Z, calculated in the spatial domain. e) the mask $m = I[x > 0]$. f) The spectrum of the mask, M. g) $M * F(X)$, calculated by direct convolution in the frequency domain. Observe this is identical to image c. h) the inverse Fourier transform of image g, which again recovers b. We plot only the magnitude of the complex-valued Fourier spectra.

X with the Fourier transform $M(x)$ of the image mask, so $\mathrm{ReLU}(X) = M(x) * X$. This is visualised in Figure 8.3.

This can be used to prove a result constraining their behaviour in frequency space, intuitively due to the fact that a convolution is a smoothing operation. Our proof relies on a particular assumption, that low frequencies dominate the input to a network in the following sense:

Definition 8.1 (Domination)**.** *We say an interval $[x, x + L]$* **dominates** *a measure $\mu(x)$ if there is more mass in the interval $[x, x + L]$ than in any other interval of equivalent size. For 2D discrete power spectra |X|, this assumption is that $\sum_u^{u+L_1} \sum_v^{v+L_2} |X_{u'v'}| \geq \sum_{u+t_1}^{u+L_1+t_1} \sum_{v+t_2}^{v+L_2+t_1} |X_{u'v'}| \forall (t_1, t_2)$*

Note that low frequency domination is a reasonable assumption for images - the typical power as a function of frequency for images and intermediate feature maps in CIFAR10 is shown in Figure 8.4, showing that the low frequencies dominate. Even if our strict domination assumption is violated, *severe* violations are rare. Using this definition, we can state a key supporting result, which is proved in Appendix C.2.

Lemma 8.2. *Let H_u be defined as above, let x, y be images, and let $v = x - y$. Assume that the power spectrum of the difference image, $|V|$, is dominated by frequencies with absolute value below a cutoff frequency u. Then* $||H_u(\mathrm{ReLU}(x) - \mathrm{ReLU}(y))|| < ||H_u(x) - H_u(y)||$.

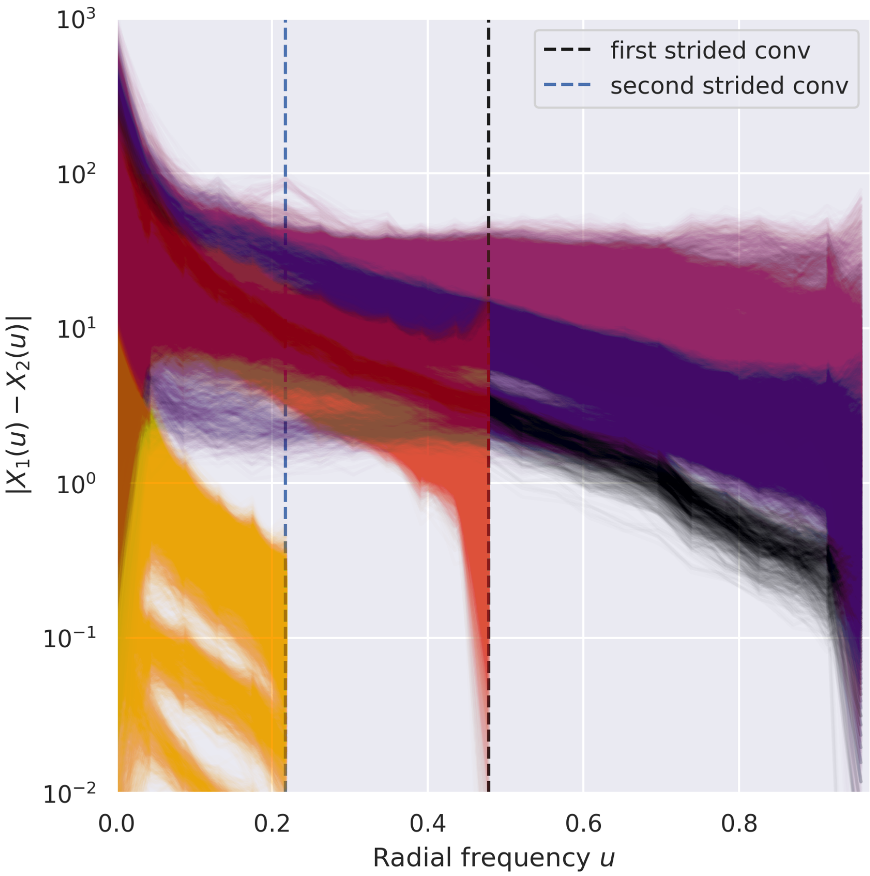


Figure 8.4: The frequency spectrum of difference images $|x_1 - x_2|$ between the feature maps of a trained neural network, for inputs x_1, x_2 randomly chosen from the CIFAR10 test set. The original image is in black, and the later feature maps are plotted in increasingly bright colours. The spectrum of every image in the sample is plotted as a transparent line. In a 2D Fourier transform, the frequency is a 2D vector $f = (f_x, f_y)$. Here, we treat all frequencies with the same radial frequency $\sqrt{f_x^2 + f_y^2}$ as equivalent. This is an empirical demonstration of our assertion that typical difference images (and feature maps) have most of their power concentrated in the low frequencies; even if our strict domination assumption is violated, it is very rarely violated strongly. Note that, after downsampling, the highest resolvable frequency is reduced, as described in Section 8.3.1. This network contains two downsampling layers, and the maximum frequency after each is shown as a dotted line.

Lemma 8.2 establishes that the ReLU contracts the low frequency component $H_u(X)$ of its input, under low-power domination. Note that the proof of this theorem uses the form of the ReLU in the frequency domain, but makes no assumptions about the form of the image mask, in order to deal with the ReLU's input dependence. We would therefore expect that the sufficient condition of full domination is likely to be fairly loose, and the ReLU may be a low-frequency contraction when this condition is violated, especially if images are 'nearly' low frequency dominant. Our results in Section 8.6 suggest that this is indeed the case. Assuming we apply spectral normalisation to the convolutions, this is also true of all other ResNet components, which allows us to use the Lemma to prove the more general result:

Theorem 8.3. *Let $f = x + g(x)$ be a convolutional residual block (i.e g is a series of convolutions, batch normalisation and ReLUs), and assume that g is regularised to be contraction ($\mathrm{Lip}(g) < 1$), and that the conditions of Lemma 8.2 hold on the input to the ReLU (i.e the difference image $x - y$ is low-frequency dominant). Then, the low-passed distances between the output are lower-bounded by the low-passed distances on the input, that is $L||H_u(x) - H_u(y)|| \leq ||H_u(f(x)) - H_u(f(y))||$ for some constant $L > 0$.*

for which a full proof is provided in Appendix C.2. This result can be extended easily to the whole network by the multiplicativity of (upper and lower) Lipschitz constants.

As stated, this theorem relates the low-frequency distances of between outputs from a ResNet and outputs from a ResNet, with no mention of dimensionality reduction. However, using Lemma 8.1, this result can be applied to ResNet blocks which use downsampling layers, as these can be re-written as a non-downsampling ResNet followed by a decimation operation. As mentioned in the previous section, a decimation operation will have no effect on the low-frequency content of the image, so the bound in Theorem 8.3 applies equally to models which include dimensionality reduction, which establishes a constraint on the feature collapse induced by practical ResNets including strided convolution layers, at least for a subset of inputs. However, we know that natural images are likely to be low-frequency dominant, and thus fall into this subset.

This theorem can be applied for any cutoff for what is considered 'low frequency', so long as the low-frequency bucket is dominant. However, the *relevant* threshold for a given network is defined by the architecture. For example, a ResNet containing 2 layers with stride 2 (and an arbitrary number of stride 1 layers) has a total downsampling factor of 4—that is, the final feature map can only resolve frequencies of $u_{\max}/4$, where $u_{\max}$ is the highest frequency resolvable in the input image. The relevant cutoff frequency is such that the operator H_u is the *identity* on the final feature map, i.e $u = u_{\max}/D$ for a network with a total decimation factor of D; that is, we need to assume that the input is dominated by frequencies which are still resolvable if downsampled to the dimensions of the final spatial feature map. This may have some implications for architecture design - if it is known empirically or through some domain knowledge what cutoff frequency is likely to be reasonable for a given domain, this could inform the amount of decimation we are willing to use in our architecture search.

It is important to discuss some limitations of these results. Firstly, the theorem only establishes a bound if the *difference* between images is low-band dominant. If $x-y$ is dominated by high frequencies, then our theorem does not apply. Extension to the full network assumes that, not only is $x-y$ low band dominant, but $f_l(x)-f_l(y)$ is low-band dominant for all intermediate feature maps l before downsampling layers. However, this assumption does appear to be true or at least a good approximation in practice; Figure 8.4 shows that the difference images between natural images and the feature representation of these images are typically concentrated in the low frequencies in trained networks.

8.5 Finding Counter-Examples

The proof in the previous section shows that the low-frequency distances between two inputs x and y will be approximately preserved by a neural mapping, given our assumptions. These conditions seem reasonable, but we may be able to find counter examples.

We know from Theorem 8.2 that examples of feature collapse must exist, but our proof of this theorem is non-constructive. We can consider a constructive procedure to demonstrate that feature collapse is impossible.

If our function $f : \mathbb{R}^n \to \mathbb{R}^m$ is continuous and differentiable, then we can consider its local Jacobian $J(x) = \nabla_x f(x) \in \mathbb{R}^m \times \mathbb{R}^n$. The value of the function around this point can be approximated by the Taylor series $f(x+t) = f(x) + J(x)t + \mathcal{O}(||t||^2)$. If we consider the singular value decomposition of the Jacobian $J = USV^\top$, there will only be m columns of V (the right singular vectors) corresponding to non-zero singular values. The remaining $n - m$ columns of V form a basis for the null space of J, that is, the set of vectors $\{v \mid Jv = 0\}$. Therefore, starting at any point x, we can take a step $t \in \{v \mid J(x)v = 0\}$ in any direction in this null space and have that $f(x) - f(x+t) = \mathcal{O}(||t||^2)$ from the Taylor expansion, since $Jt = 0$. We can repeat this process iteratively with n steps t_i of length ϵ, such that each t_i lies in the null space of the Jacobian $J(x + \sum_1^{i-1} t_i)$ to generate, starting from a point x_0, a counterpart $x_n = x_0 + \sum_{i=1}^n t_i$.

We can freely choose t_i such that $\langle t_i, t_j \rangle \geq 0$ for all i, j, so that the steps do not cancel each other out. In this case, the total distance covered by the trajectory, $||x_0 - x_n|| = ||\sum_{i=1}^n t_i|| \simeq \mathcal{O}(\epsilon n)$, whereas the corresponding change in feature space is $||f(x_0) - f(x_n)|| \simeq \mathcal{O}(n\epsilon^2)$. It is clear from this that by adding more steps to this trajectory and making ϵ arbitrarily small while holding $n\epsilon = C$ constant, the effect on the output can be made arbitrarily small ($\mathcal{O}(C\epsilon)$) while the length of the trajectory remains $\mathcal{O}(C)$, so we obtain a recipe for finding two points x_0 and x_n with an arbitrarily small distance in feature space. This argument is a little less mathematically precise than our proof of Theorem 8.2, but it does suggest a constructive algorithm for finding *examples* of feature collapse, which is outlined in algorithm 8.1

However, this procedure is extremely slow, as it requires the computation of the full SVD of the Jacobian at every step.

Algorithm 8.1 Given image x_0, find an image x such that $f(x_0) \simeq f(x)$ and $||x_0 - x|| > 0$.

Input: Input image x_0, feature mapping f, step size η, number of steps n, distribution over unit vectors D
Output: Example image x
$y_0 \leftarrow f(x_0)$
$r \leftarrow D()$
$x \leftarrow x_0 + \eta v$
for $i \in 0..n-1$ **do**
  $r \leftarrow D()$ ▷ Sample random step direction
  $J \leftarrow \nabla_x f(x)$ ▷ find the Jacobian of the feature mapping
  $\_, \_, V \leftarrow \text{svd}(J)$ ▷ Compute the (compact) right singular vectors of the Jacobian (i.e those with non-zero singular values)
  $u \leftarrow r - \sum_i V[:, i](\langle V[:, i], r\rangle)$ ▷ Project the step direction into the null space of the Jacobian
  $x \leftarrow x + \eta u$ ▷ Take a step along the level set
end for

To avoid fully computing the SVD of the Jacobian, we consider taking steps which hold the norm of $||f(x) - y_0||$ constant, rather than the function $f(x)$ directly, as this only requires computing a vector gradient, exploiting the fact that the level sets of a function are locally orthogonal to the gradient, to achieve a similar aim of choosing steps that keep $f(x)$ approximately constant. This is described in Algorithm 8.2, which can produce counter-examples in a reasonable amount of time.

Algorithm 8.2 Given image x_0, find an image x such that $f(x_0) \simeq f(x)$ and $||x_0 - x|| > 0$.

Input: Input x_0, function f, step size η, number of steps n, distribution over unit vectors D
$y_0 \leftarrow f(x_0)$
for $i \in 0..n-1$ **do**
  $v \leftarrow D()$ ▷ Sample random step direction
  $g \leftarrow \nabla_x (||f(x) - y_0||)$ ▷ find the gradient of the distance from the target output
  $u \leftarrow v - \frac{\langle v,g\rangle}{\langle g,g\rangle} g$ ▷ Project the step to be orthogonal to the gradient
  $x \leftarrow x + \eta u$ ▷ Take a step along the level set
end for
return x

We can influence the properties of the generated example x by changing the distribution of v. It is simple to, for example, choose v to be band limited, generating examples where $x_0 - x$ is either low- or high-frequency dominant. We investigate both of these below.

8.6 Experiments

8.6.1 Verifying our theoretical analysis

There are two key claims in Section 8.4 we want to verify empirically. These are 1) do the conditions of Lemma 8.2 hold (i.e are the images and feature maps low frequency dominant in the sense of this lemma)? And 2) do the *conclusions* of Theorem 8.3 hold, that is, is $||H_u(g(x)) - H_u(g(y))|| < ||H_u(x) - H_u(y)||$ for the inputs to each residual block?

The conditions of Lemma 8.2 are sufficient for low-frequency distances to be preserved but we regard them unlikely to be necessary conditions. This is because, as discussed in Section 8.4, our domination assumption is a sufficient condition for the ReLU to act as a low-frequency contraction even under a worst-case analysis, which we use to deal with the non-linearity of the ReLU. As a result, when testing on natural images, it seems reasonable to expect that the ReLU may act as a low-frequency contraction even if the domination assumption is not met strictly, particularly if, as shown in Figure 8.4, the images are fairly close to being low-frequency dominant.

In order to investigate this, we train a Wide ResNet (Zagoruyko and Komodakis, 2016) (with minor changes discussed in Appendix C.4) on MNIST (LeCun et al., 1998b), FashionMNIST (Xiao et al., 2017) and CIFAR-10 (Krizhevsky, Hinton, et al., 2009). We check the domination of an image or feature map directly from its Fourier transform, empirically checking claim 1). To verify that the residual connection is a contraction on the low-frequencies, we take a mini-batch of data $\{x_i \mid i \in 1..m\}$, and consider pair-wise distances before and after the residual connection, calculating the proportion of the batch for which $||g(H_u(x_i)) - g(H_u(x_j))|| < ||H_u(x_i) - H_u(x_j)||$, checking claim 2). If we cannot find violations on natural image datasets, then this suggests that low-frequency distances are preserved *between natural images.*

These results are shown for FashionMNIST in Table 8.1. We also carried out the same experiment on MNIST and CIFAR10, these results are available in Appendix C.3.1 and consistent with Table 8.1. Theorem 8.3 is supported remarkably well by the evidence - for $\mathrm{Lip}(g) < 1$ we are unable to find *any* counterexamples in the image datasets tested. We also find that the low-frequency domination condition (Lemma 8.2) holds on a subset of the data, but not on a sufficient proportion of images to fully explain the adherence to Theorem 8.3. Again, there are theoretical reasons to expect this to be true, as our proofs use worst case analysis, and so on natural images where the conditions are not violated strongly it would not be unreasonable to expect the results to hold.

We are also interested in relaxing the assumption that the convolution layers are strict contractions, with $\mathrm{Lip}(g) < 1$. This has been widely used in prior work on 'bilipschitz' regularisation in practice, including in the preceding chapter, despite the lack of theoretical guarantees in this case. Indeed, when we increase $\mathrm{Lip}(g)$ above 1, we start to see violations of Theorem 8.3 in trained models, though this remains rare.

Interestingly, we observe that models appear to converge to satisfy our condition during training; an example of this dynamic is shown in Figure 8.5. This shows that, while models with spectral coefficients are contractions on the low frequencies throughout training, as our theory would predict, models with higher coefficients violate our theorem early in training, but appear to become contractions on the low frequencies later in training. We do not have a theoretical explanation for why this should happen, but it is in agreement with the empirical observation in prior work that distance aware learning with spectral normalisation coefficients above 1 have performed well, such as the results presented in the previous chapter, despite the lack of guarantees that they should avoid feature collapse.

8.6.2 Artificially finding counter-examples

We use Algorithm 8.2 to find counter-examples to our assumption that feature collapse is rare on natural image datasets. We can find such examples by explicit optimisation, as shown in Figure 8.6 by the line designated "all freq". This does *not*

	Lip(g)	x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Lemma 8.2	0.9	0.86 ± 0.00	0.18 ± 0.04	0.24 ± 0.04	0.48 ± 0.04	0.47 ± 0.03
	0.95	0.86 ± 0.00	0.18 ± 0.04	0.25 ± 0.02	0.48 ± 0.03	0.46 ± 0.01
	0.99	0.86 ± 0.00	0.18 ± 0.05	0.25 ± 0.04	0.51 ± 0.02	0.48 ± 0.01
	3.0	0.86 ± 0.00	0.24 ± 0.03	0.28 ± 0.04	0.53 ± 0.04	0.47 ± 0.02
	6.0	0.86 ± 0.00	0.24 ± 0.03	0.28 ± 0.02	0.53 ± 0.02	0.47 ± 0.02
	9.0	0.86 ± 0.00	0.25 ± 0.01	0.28 ± 0.04	0.54 ± 0.04	0.47 ± 0.01
	no	0.86 ± 0.00	0.24 ± 0.03	0.25 ± 0.03	0.53 ± 0.03	0.47 ± 0.01
Theorem 8.3	0.9	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.95	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.99	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	3.0	n/a	0.98 ± 0.04	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	6.0	n/a	0.95 ± 0.11	0.92 ± 0.04	1.00 ± 0.00	1.00 ± 0.00
	9.0	n/a	1.00 ± 0.01	0.85 ± 0.12	1.00 ± 0.00	1.00 ± 0.00
	no	n/a	1.00 ± 0.00	0.86 ± 0.07	1.00 ± 0.00	1.00 ± 0.00

Table 8.1: Results of empirically checking the conditions of Lemma 8.2 (low frequency domination) and Theorem 8.3 (low frequency contraction) on FMNIST (results on other datasets can be found in Appendix C.3.1). Numbers are the proportion of the dataset for which we found the condition to hold exactly; so 1 means no violations were found, 0.5 means it was true for half the inputs tested, etc. We report means and standard error over 25 seeds and use a WideResNet with depth 10 and widen factor of 1 for these datasets.

disprove Theorem 8.3 - an important assumption of our theorem is that not just $x - y$ is low-band dominant, but $f_l(x) - f_l(y)$ for all intermediate layers l. While we observed that this is generally true, it does not preclude the possibility that there could exist pairs of inputs whose distance is low band dominant but where this is no longer true for the distance between feature maps, and the existence of counter examples suggests that this is possible.

This suggests that proving a significantly stronger theoretical guarantee than the one provided in this paper is unlikely to be possible unless additional constraints on the model behaviour are added. While empirically low-frequency dominant input often produces low-frequency dominant feature maps, as shown in Figure 8.4 and as assumed in our proofs, it is not structurally *necessary* that this is the case, as by explicit optimisation, we are able to find low frequency inputs which induce high-frequency dominant feature maps, allowing feature collapse to

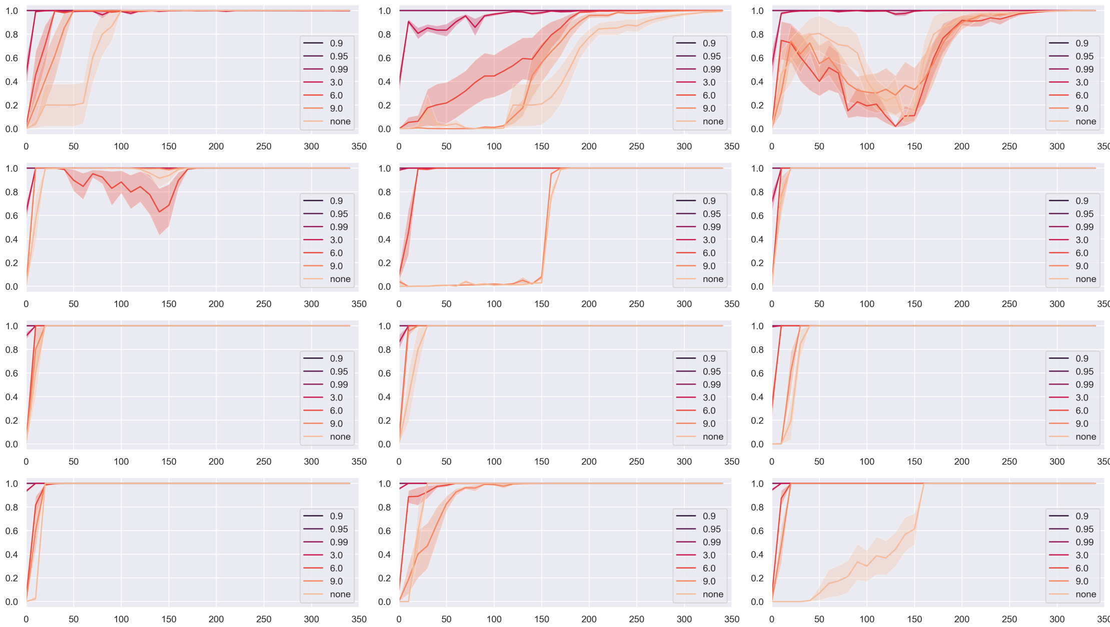


Figure 8.5: The network increasingly satisfies our checks of Theorem 8.3 as training progresses, with nearly 100% match later in training, even for coefficients above 1, while models with coefficients less than one satisfy it throughout, as our theory would predict. We examine the blocks of a Wide ResNet trained on CIFAR-10, for various values of the spectral normalisation coefficient (Lip(g)). We plot the mean and standard error across 5 seeds.

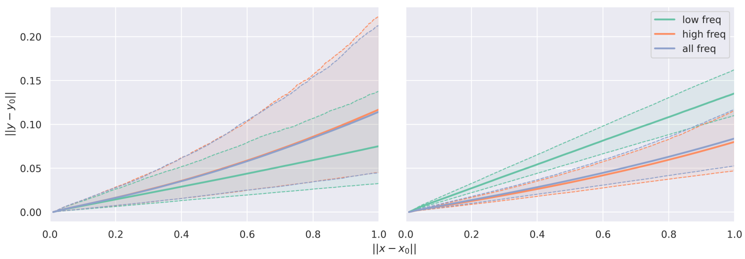


Figure 8.6: Distance in feature space ($||y - y_0||_2$) against distance in image space ($||x - x_0||_2$) during a run of Algorithm 8.2. Left; for a standard model, right; for a model where we low-pass-filter the feature maps between residual blocks, enforcing the domination assumption. We show results where the trajectory is restricted to either low band or high band perturbation and unrestricted. On each, we show the mean and 10th and 90th percentile across 1000 images. We provide exact values at convergence in Appendix C.3.3.

occur in the models studied.

To further verify that this is a result of our *assumption* being violated, and not the content of our theorem itself, we experiment with *enforcing* that the feature maps are low-band dominant by inserting low-pass filters after every residual block.

The results can be found in Figure 8.6 with the line “low freq”. In these models, as expected, we find that we are unable to find low-frequency perturbations which do not significantly increase the feature space distance, unlike for unconstrained or high frequency perturbations. Adding these filters on the feature maps, however, is very restrictive, and these models are not competitive in terms of classification accuracy. We leave whether we can enforce the conditions of our theorem in a less restrictive way to future work. For completeness we show that keeping only the high frequencies (“high freq”) has similar behaviour to keeping all.

8.7 Conclusion and Limitations

We have demonstrated that, under mild assumptions, regularised residual networks approximately preserve distances on the low-passed portion of their input. Both our assumptions and results appear to hold in practice on natural images. Even when our assumptions are relaxed, for example by training with a spectral norm coefficient larger than one, the conclusions of our theoretical analysis – that low-frequency distances should be preserved – continue to hold empirically, and our theoretical proof likely provides some insight into why these architectures and regularisation schemes seem to avoid feature collapse on datasets in practice. This puts some of the mathematically dubious claims about avoiding feature collapse used to originally motivate the work presented in the previous chapter on a more solid theoretical basis, which is desirable as the results already presented have shown that this is potentially a promising direction.

Our work has important implications for future methods based on feature space distances. Theorem 8.2 entails that avoiding feature collapse completely is impossible unless our models are dimensionality preserving, something which past methods have not taken into account. This also shows that existing claims to be “distance sensitive” or “preserve distances” must be treated cautiously, as well as showing that preventing feature collapse entirely is not a useful design goal for models which perform dimensionality reduction.

On the other hand, we show that it is possible to be "distance sensitive" on a *useful* subspace of the input, and that existing models already achieve this to a degree. In particular, distances are preserved in a restricted sense on a particular subset of the input space (low-frequency dominant images whose intermediate representations stay low-frequency dominant), which we show makes up a non-trivial portion of typical test datasets for image models. This frequency domain perspective could prove useful for future work.

While the low-frequency domain is a natural subset to focus on given its mathematical links to convolutional models, it is possible that other 'weaker' distance preservation conditions, which do not violate Theorem 8.2 could be developed in the future. For instance, one could consider other ways to characterise the data manifold, as a notable conceptual drawback of the frequency-based guarantees presented here is that they are not data dependent in any way. This could further inform future architecture designs.

One particularly interesting empirical result in this paper is the observation in Figure 8.5 that models with a spectral coefficient larger than one still appear to preserve distances on low frequency projections empirically. Our proof construction relies on the Lipschitz bound of the residual connection, but while we do observe that models with a Lipschitz constraint violate our theorem during training, as we would expect from our theory, they tend to be low-frequency contractions empirically increasingly as training progresses. While this does not show that our mathematical analysis is wrong, it is not particularly what you would expect from our theoretical argument, which would generally predict that unconstrained models would violate this condition. This may suggest that, while the Lipschitz constant of the residual connection plays a crucial role in our theoretical proof, it is less important in practice than we might expect. While it is generally found to be an important component of the model, for instance, in the previous chapter and in the ablation study in Liu et al. (2020), in the sense that removing spectral normalisation hurts performance, it is hard to conclude from this experiment that spectral normalisation in these models is working entirely because of the theoretical bound presented in this chapter. We think there are number of future directions based on this observation which could be interesting. Firstly, prior practical work typically does not use a Lipschitz bound lower than one because this makes training much harder. This figure suggests that it is at least a possibility that this may be because the Lipschitz bound affects the training *dynamics*, rather than it being impossible for a trained model to have a lower Lipschitz constant. This seems somewhat unlikely, but it does suggest varying the way the Lipschitz bound is enforced during training - perhaps training an unrestricted model and gradually decreasing the Lipschitz constant towards the end of training rather than using a constant coefficient throughout could allow models with good accuracy and Lipschitz bounds sufficient to apply the theoretical proofs here.

The other possibility is that there is an alternative or complementary phenomenon explaining the success of this architecture. While we think the frequency space perspective outlined in this chapter is informative, our proof here is based on an attempt to extend the standard proof that a Lipschitz coefficient below one enforces bilipschitzness to networks which include dimensionality reduction. In light of this observation, it is possible that this focus on the Lipschitz constant as the crucial element is misleading, and perhaps a proof could be constructed based on some other feature of the architecture. It is notable that the Lipschitz bound on the residual connection is fairly crude and makes minimal assumptions about the architecture, and perhaps an analysis that made fewer such assumptions could predict observed behaviour more accurately. However, we leave this for future work to investigate.

Despite these caveats, our theoretical scheme provides a partial theoretical justification for the continued use of the spectral normalisation scheme for preventing feature collapse. It is undeniably a step forward from claims that the architecture works by enforcing a bilipschitz constraint, which as we have seen is impossible. Together with the empirical success as we demonstrated in Chapter 7, provides a useful foundation for future work on this model family which we hope will be useful for future research.

For this is the truth: I have departed from the house of the scholars, and the door have I also slammed behind me.

Friedrich Nietzsche, *Thus Spoke Zarathustra*

9 Conclusions

9.1 Possible Directions for Future Research

In this section we briefly outline some possible directions for future work which we think follow naturally from the work in this thesis, outline possible issues, and briefly describe tentative work taken towards these directions by the author, sketching potential pitfalls for future researchers on these directions.

9.1.1 Scaling DUE models

An obvious direction for future work is, given the theoretical motivation and performance we have described in the preceding two chapters, scaling up the regularised deep kernel model described in Chapter 7. There are some potential barriers to doing this effectively, but in principle this should be fairly readily approached with current tools and techniques. One potential issue is the increased number of features, due to larger images, and also the very large number of classes in larger image datasets. Even with just a few inducing points per class, this would mean that the number of inducing points required, and therefore the size of the kernel matrix which needs to be inverted in order to train an approximate GP using the method used in Chapter 7, does not scale particularly well, and may hinder attempts to scale up training.

One interesting, though less elegant, possibility is to simply use regularisation methods like those in Part III to train a standard softmax model, then use its frozen feature space to train a distance sensitive model on top. This approach was shown to be fairly promising in the preliminary work of Mukhoti et al. (2021), and potentially fits well into the increasingly popular paradigm of using large, pretrained 'foundation' models (Bommasani et al., 2021) as the basis for multiple tasks. Studying whether feature collapse and the issues identified more broadly in this thesis are shared by models trained using this methodology, which are increasingly important in practice, would be an important direction for future work.

9.1.2 Further progress on feature collapse

While we think that the theory outlined in Chapter 8 is a reasonable and useful attempt at understanding why the regularisation scheme used for the models in Part III was effective, it is clearly not a complete theory. In Section 8.7, we outlined some drawbacks of the framework, in particular its lack of really strong guarantees and its inability to explain why relaxing the Lipschitz constraint appears to still be an effective technique empirically.

One possible way forward is to aim at avoiding feature collapse on a more limited subset. Chapter 8 chose a fixed, data-independent subset of input space (low-frequency projections). While natural images do tend to be low-frequency dominant, it remains the case that this guarantee was chosen a-posteriori for mathematical convenience in convolution networks specifically rather than being an explicit design goal. An alternative possibility is to note that the space of images we are really interested in - natural images, natural text etc. - do not really 'fill' the Euclidean input space of a neural network, but form a lower-dimensional manifold within it. While it is impossible to mathematically characterise this, it may be possible to formulate regularisation schemes aimed at avoiding feature collapse on this manifold, as defined by data sampling, rather than on any possible input. These could potentially make use of unlabelled data to define this manifold. There is potentially a link with methods such as contrastive predictive coding (Oord et al.,

2018) that utilise very large amounts of untrained data. It is not clear that such objectives suffer from the same issues around encouraging feature collapse discussed in Section 6.3, and experiments to test whether these alternative training paradigms suffer from similar issues to those identified in the course of this thesis in supervised classification models would be an interesting direction for work in the future.

9.1.3 Making use of the feature space GP

One of the potentially exciting things about getting models like that described in Part III to work better is that there are many potential use-cases that having a model like a GP as a decision function makes easier. For instance, in a GP quantities like the marginal likelihood and joint entropy (of the feature space model) are easy to evaluate compared to a Bayesian neural network, if the feature inputs can be taken as constant. If the regularisation scheme presented in Part III is powerful enough that these quantities are informative, this could be an extremely useful practical advantage over Bayesian neural networks in particular applications, and could allow interesting methodological progress.

For example, in a Gaussian process, the mutual information between the input and output, as used in Chapter 2 can in some cases be analytically evaluated, depending on the likelihood. As the mutual information is frequently used as an acquisition function in active learning – see, for instance, Houlsby (2014), Kirsch et al. (2019), and Walmsley et al. (2019) – it is possible that this could enable much lower-variance and computationally efficient evaluation of these quantities, which could be advantageous when deploying active learning in practice.

Another interesting application is the use of the marginal likelihood for model selection. While the marginal likelihood of the GP in feature space is not a true evaluation of the Bayesian marginal likelihood, which ought to also accommodate uncertainty over the weights of the network, if it was a reasonable proxy for this quantity it could still be extremely useful. For instance, if this was informative for model selection, it could be used as a criterion for model selection or hyperparameter search. Simple proxies such as the sum over training losses (Lyle et al., 2020) have been demonstrated to be potentially useful, so it is not unreasonable to investigate whether the marginal likelihood of this kind of model would be informative in this regard.

One potentially interesting use of the marginal likelihood is to learn invariances, as in Wilk et al. (2018). In this paper, the marginal likelihood is used to learn a parameterised invariance, by incorporating averaging over a parameterised transformation of the data into the kernel. However, on a more negative note, the author of this thesis did conduct some preliminary experiments along these lines, attempting to use the marginal likelihood of a feature-space GP to fit an invariance applied to the input of the neural network. Unfortunately, we did not find that the marginal likelihood in feature space was informative enough for this method to work; in this case, the pitfalls of deep kernel learning identified by Ober et al. (2021), namely that the loss encourages the model to collapse input points very close to each other, can still mean that the marginal likelihood in feature space is insufficiently informative.

One way to think about this is, even if the model is dimensionality preserving, the bilipschitz constraints enforced by the model are still fairly loose for models of a reasonable size. For instance, consider a model with a depth of 10 residual blocks and a spectral norm coefficient of 0.5, which is much lower than typically used in the experiments of Chapter 7. In this case, the lower Lipschitz bound of the network, applying Theorem 8.1, is $(1-0.5)^{10}$ and the upper bound is $(1+0.5)^{10}$. As such, while absolute feature collapse is prevented in this case, in practice the ratio between two equal distances in input space can be as high as $\left(\frac{1+0.5}{1-0.5}\right)^{10} \simeq 6 \times 10^4$, so several orders of magnitude, which is likely sufficient for the effect of the feature extractor on the marginal likelihood to dwarf any augmentations applied to the input. While it appears that the controlling of the Lipschitz constant is sufficient to control the behaviour of the model out of distribution, this is a reason to be pessimistic that measures like the marginal likelihood in feature space might not be particularly useful, especially when used as targets for explicit optimisation.

9.1.4 Language & Terminology

An important insight that the author, at least, took away from working on this thesis is the importance of being careful with *terminology*, especially when framing the goals that we are aiming at. In particular, with capsules, the focus on capturing 'objects', without a clear operational definition of what we meant by capturing objects, led to something of a quagmire. Motivation for the capsule model in previous work, and to a degree in our own, was in hindsight based at least in part on conflating a common-language term - object - which is vague, expansive and poorly defined, with a notion of a geometric arrangement of fixed pixel templates which bore little, if any, relationship to the definition we had in mind. In particular, there was a lack of empirical tests for whether a model had successfully learnt to represent objects.

While we have not addressed this directly in this thesis, it is worth considering whether other aims we have worked towards may follow a similar pattern. In particular, an important goal throughout this thesis has been getting 'reasonable' uncertainty. Unlike objects, frequently we have had an *implied* operational definition for this; for instance, reasonable uncertainty has meant, in practice, having high uncertainty on out-of-distribution samples, allowing deferring predictions to an expert or registering an image as adversarial. These are not unreasonable operational definitions, but they are potentially more narrow than what we have in mind when we say reasonable uncertainty, and we should be careful about assuming that a model with reasonable uncertainty in one sense always behaves the way we want in other situations. We don't think this is a particularly uncommon situation - intelligence and learning in general are difficult concepts to pin down despite their centrality to the field, and occasional missteps are to be expected. Arguments over the appropriate operational definition of machine intelligence date back at least to Turing (1950). Nevertheless, the need to think carefully about what exactly our model structure or benchmarks are measuring, and being cautious about using potentially vague terms without careful consideration, has not diminished in importance since.

9.2 Conclusion

In this thesis, we were aiming to answer the question of whether applying the standard Bayesian deep learning toolkit was the best way to obtain uncertainty estimation in domains where we need to use deep models to get state of the art performance, but want to quantify the uncertainty of our model as accurately as possible. Examples of such domains are, for example, in accelerator control for the production of high intensity X rays (Convery, **Smith**, Gal, and Hanuka, 2021), where it is important to have bounds on the possible emission of the system to avoid damaging the samples of downstream users of the light source, which may be expensive or impossible to replicate. Another is if uncertainty is used as part of the learning algorithm itself, such as in active learning (Walmsley, **Smith**, Lintott, Gal, Bamford, Dickinson, Fortson, Kruk, Masters, Scarlata, Simmons, Smethurst, and Wright, 2019).

We have presented some research, in Part I of this thesis, arguing that the standard Bayesian deep learning approach of learning distributions over weights, while useful, inherits many of the flaws of standard deep learning when local approximations are used; it is vulnerable to adversarial examples, and prone to highly confident extrapolation unsupported by data, meaning that the uncertainty estimates produced by Bayesian deep learning methods are not entirely reliable.

In the second and third parts of this thesis, we explored some alternatives to this, by imposing additional structure on the deep model. The second part presented a generative formulation of *capsule networks*, which aim to enforce equivariance by directly representing objects as structured arrangements of geometrically interrelated parts. While this idea appeared promising, our work in attempting to make these ideas formal and explore them using a rigorous probabilistic formulation suggests that the attempt to split the world into ‘objects’ is poorly defined, without further assumptions. The implicit assumption made in capsule networks - that ‘objects’ learned by a structured model trained on an arbitrary classification or reconstruction task will map onto semantic objects - is not well supported by our research into this.

The two key ideas of capsules, however - modelling the relationships between objects, and equivariance - remain potentially promising directions for future research. Equivariance can probably be better approached by enforcing it explicitly, utilising group theory as in Weiler and Cesa (2019).

The concept of 'objects', as we have seen, is poorly defined a priori. However, it is possible that the focus on objects like this could still be useful, if these are defined more concretely by an attendant task. For instance, if rather than attempting to learn object purely from perception, an object-focused system was used to learn to manipulate object arrangements with a robot arm – where an object now has an *operational* definition – an ability to explicitly represent objects may provide some advantages. We feel, though, that a major takeaway from our work on this subject is that practitioners and researchers should ask what the objects in a system *do*. Without a more well-defined operational definition of object, we are not optimistic about the prospects of further work on object focussed models.

In the final section of this thesis, we introduce a particular form of 'bilipschitz' models, demonstrate their empirical performance, and examine the theory underpinning the regularisation scheme more closely. In contrast to the work presented here on capsules, we do feel that this kind of model represents a plausible alternative to standard approaches to uncertainty in deep learning, with considerable empirical advantages in terms of processing time and performance compared to standard networks or ensembles in the benchmarks tested. Future work remains to be done on the scalability of these methods to larger datasets. In addition, there remain gaps in the theoretical understanding of these methods; while we have presented some preliminary work on this subject, it is not clear that this is the whole story, and it does not present a particularly clear direction on how to strengthen the control of feature collapse in this kind of model. Scaling this kind of GP model to higher dimensional feature spaces is another possible difficulty.

This family of methods does have some potential areas which might be fruitful for future research. In particular, using classical probabilistic methods in the feature space could be used to design practical methods, reviving some of the initial promise of deep kernel learning.

More work could be done on generalising the idea of avoiding feature collapse to non convolutional architectures, or strengthening the theoretical guarantees outlined in Chapter 6. For example, transformer architectures have become increasingly important in recent years especially in natural language processing; avoiding feature collapse could be an important goal for the sort of 'foundation models' as proposed in Bommasani et al. (2021).

Going forward, we think, despite the caveats outlined, the methods described in Part III are part of the potential toolbox available to practitioners who are interested in uncertainty quantification in deep learning. However, in answer to the fundamental question of this thesis, standard Bayesian deep learning, despite the faults described in this work, remains an important and flexible tool, and probably the best choice in many practical scenarios, though we think practitioners should consider the models described in part III as a plausible alternative.

In addition, though, we think that the insights contained above show that practitioners should avoid the temptation to treat their models as a 'black box' to which uncertainty can be added, but to be aware of the interplay between inference and the structure of the model itself. There is often a temptation when thinking about uncertainty to ignore the complexities of the model architecture, but these are crucial to consider as well. This is particularly apparent in Part III, where modifying the structure of the non-probabilistic component of the model has a central role in regulating the uncertainty behaviour of the whole. This is perhaps not a particularly new lesson - in many ways, it is central to the philosophy of Bayesian inference in general that priors are both influential and unavoidable - but the age of deep learning has not diminished its importance.

Computer science is a terrible name for this business: first of all it's not really a science, more a kind of engineering ... Its also not very much about computers.

Hal Abelson

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.

Amersfoort, Joost van et al. (2020). "Uncertainty Estimation Using a Single Deep Deterministic Neural Network". In: *International Conference on Machine Learning.*

Amersfoort, Joost van et al. (2021). "Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression". In: *arXiv preprint arXiv:2102.11409.*

Aronszajn, Nachman (1950). "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3, pp. 337–404.

Baydin, Atilim Gunes et al. (2018). "Automatic differentiation in machine learning: a survey". In: *Journal of Marchine Learning Research* 18, pp. 1–43.

Behrmann, Jens et al. (2019). "Invertible residual networks". In: *International Conference on Machine Learning*, pp. 573–582.

Bingham, Eli et al. (2018). "Pyro: Deep Universal Probabilistic Programming". In: *Journal of Machine Learning Research.*

Bishop, Chris (2006). "Introduction". In: *Pattern Recognition and Machine Learning.* Springer.

Blundell, Charles et al. (2015). "Weight Uncertainty in Neural Network". In: *International Conference on Machine Learning*, pp. 1613–1622.

Bommasani, Rishi et al. (2021). "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258.*

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.*

Brouwer, Luitzen EJ (1911). "Beweis der invarianz des n-dimensionalen gebiets". In: *Mathematische Annalen* 71.3, pp. 305–313.

Bubeck, Sébastien and Mark Sellke (2021). "A universal law of robustness via isoperimetry". In: *Advances in Neural Information Processing Systems* 34.

Burgess, Christopher P et al. (2019). "Monet: Unsupervised scene decomposition and representation". In: *arXiv preprint arXiv:1901.11390.*

Carlini, Nicholas and David Wagner (2017a). "Adversarial examples are not easily detected: Bypassing ten detection methods". In: *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 3–14.

— (2017b). "Towards evaluating the robustness of neural networks". In: *Security and Privacy (SP), 2017 IEEE Symposium on.* IEEE, pp. 39–57.

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.

Choromanska, Anna et al. (2015). "The loss surfaces of multilayer networks". In: *Artificial Intelligence and Statistics*, pp. 192–204.

Convery, Owen et al. (2021). "Uncertainty quantification for virtual diagnostic of particle accelerators". In: *Physical Review Accelerators and Beams* 24 (7), p. 074602. URL: `https://link.aps.org/doi/10.1103/PhysRevAccelBeams.24.074602`.

Cox, Richard T (1946). "Probability, frequency and reasonable expectation". In: *American journal of physics* 14.1, pp. 1–13.

De Finetti, Bruno (1937). "Foresight: Its Logical Laws, Its Subjective Sources". In: *Studies in Subjective Probability.*

Diaconis, Persi, Susan Holmes, and Richard Montgomery (2007). "Dynamical bias in the coin toss". In: *SIAM review* 49.2, pp. 211–235.

Dong, Yinpeng et al. (2018). "Boosting adversarial attacks with momentum". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193.

Druţu, Cornelia and Michael Kapovich (2018). *Geometric group theory.* Vol. 63. American Mathematical Soc.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository.* URL: `http://archive.ics.uci.edu/ml`.

Elson, Jeremy et al. (2007). "Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization". In: *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS).* Association for Computing Machinery, Inc. URL: `https://www.microsoft.com/en-us/research/publication/asirra-a-captcha-that-exploits-interest-aligned-manual-image-categorization/`.

Eslami, SM Ali et al. (2016). "Attend, infer, repeat: Fast scene understanding with generative models". In: *Advances in Neural Information Processing Systems*, pp. 3225–3233.

Farquhar, Sebastian, Michael Osborne, and Yarin Gal (2020a). "Radial Bayesian Neural Networks: Beyond Discrete Support In Large-Scale Bayesian Deep Learning". In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics.*

Farquhar, Sebastian, **Lewis Smith**, and Yarin Gal (2020b). "Liberty or depth: Deep Bayesian neural nets do not need complex weight posterior approximations". In: *Advances in Neural Information Processing Systems.*

Fawzi, Alhussein, Omar Fawzi, and Pascal Frossard (2018). "Analysis of classifiers' robustness to adversarial perturbations". In: *Machine Learning* 107.3, pp. 481–508.

Feinman, Reuben et al. (2017). "Detecting Adversarial Samples from Artifacts". In: *arXiv preprint arXiv:1703.00410.*

Foong, Andrew YK et al. (June 2019). "'In-Between'Uncertainty in Bayesian Neural Networks". en. In: *Workshop on Uncertainty and Robustness in Deep Learning.* arXiv: `1906.11537 [cs, stat]`.

Gal, Yarin (2016). "Uncertainty in deep learning". PhD thesis. PhD thesis, University of Cambridge, p. 3.

Gal, Yarin, Yutian Chen, and Zoubin Ghahramani (2015). "Latent Gaussian processes for distribution estimation of multivariate categorical data". In: pp. 645–654.

Gal, Yarin and Zoubin Ghahramani (2016). "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". In: *International Conference on Machine Learning.* Vol. 48, pp. 1050–1059.

Gal, Yarin, Jiri Hron, and Alex Kendall (2017). “Concrete dropout”. In: *Advances in neural information processing systems* 30.
Gal, Yarin and Lewis Smith (2018). “Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with Bayesian neural networks”. In: *arXiv preprint arXiv:1806.00667*.
Gardner, Jacob R et al. (2018). “GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration”. In: *Advances in Neural Information Processing Systems*.
Gilmer, Justin et al. (2018). “Adversarial spheres”. In: *arXiv preprint arXiv:1801.02774*.
Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy (2014). “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572*, pp. 1–11. arXiv: `1412.6572`.
Gouk, Henry et al. (2021). “Regularisation of neural networks by enforcing lipschitz continuity”. In: *Machine Learning* 110.2, pp. 393–416.
Graves, Alex (2011). “Practical variational inference for neural networks”. In: *Advances in Neural Information Processing Systems*, pp. 2348–2356.
Greff, Klaus et al. (2019). “Multi-object representation learning with iterative variational inference”. In: *International Conference on Machine Learning*. PMLR, pp. 2424–2433.
Harris, Charles R et al. (2020). “Array programming with NumPy”. In: *Nature* 585.7825, pp. 357–362.
Hayou, Soufiane, Arnaud Doucet, and Judith Rousseau (2019). “On the impact of the activation function on deep neural networks training”. In: *International conference on machine learning*. PMLR, pp. 2672–2680.
He, Kaiming et al. (2015a). *Deep residual learning for image recognition. CoRR abs/1512.03385 (2015)*.
— (2015b). “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.
Hein, Matthias and Maksym Andriushchenko (2017). “Formal guarantees on the robustness of a classifier against adversarial manipulation”. In: *Advances in neural information processing systems* 30.
Hensman, James, Alexander Matthews, and Zoubin Ghahramani (2015). “Scalable variational Gaussian process classification”. In: *JMLR*.
Hill, Jennifer L (2011). “Bayesian nonparametric modeling for causal inference”. In: *Journal of Computational and Graphical Statistics* 20.1, pp. 217–240.
Hinton, Geoffrey and Drew van Camp (1993). “Keeping Neural Networks Simple by Minimizing the Description Length of the Weights”. In: *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory*.
Hinton, Geoffrey E, Zoubin Ghahramani, and Yee Whye Teh (2000). “Learning to parse images”. In: *Advances in neural information processing systems*, pp. 463–469.
Hinton, Geoffrey E, Alex Krizhevsky, and Sida D Wang (2011). “Transforming auto-encoders”. In: *International Conference on Artificial Neural Networks*. Springer, pp. 44–51.
Hinton, Geoffrey E, Sara Sabour, and Nicholas Frosst (2018). “Matrix capsules with EM routing”. In: *International conference on learning representations*.
Hochreiter, Sepp and Juergen Schmidhuber (1997a). “Flat Minima”. In: *Neural Computation* 9.1, pp. 1–42.

Hochreiter, Sepp and Jürgen Schmidhuber (1997b). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.
Houlsby, Neil (2014). “Efficient Bayesian active learning and matrix modelling”. PhD thesis. University of Cambridge.
Huang, Wenbing et al. (2015). “Scalable gaussian process regression using deep neural networks”. In: *Twenty-fourth international joint conference on artificial intelligence.*
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. arXiv: `1502.03167`.
Jang, Eric, Shixiang Gu, and Ben Poole (2016). “Categorical reparameterization with gumbel-softmax”. In: *arXiv preprint arXiv:1611.01144*.
Jaynes, Edwin (Apr. 2003). *Probability Theory: The Logic of Science*. Cambridge University Press.
Jesson, Andrew et al. (2020). “Identifying Causal-Effect Inference Failure with Uncertainty-Aware Models”. In: *Advances in Neural Information Processing Systems* 33.
Johnson, Justin et al. (2017). “Clevr: A diagnostic dataset for compositional language and elementary visual reasoning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910.
Kahneman, Daniel and Amos Tversky (1979). “Prospect Theory: An Analysis of Decision under Risk”. In: *Econometrica* 47.2, pp. 263–292.
Kim, Hyunjik, George Papamakarios, and Andriy Mnih (2021). “The lipschitz constant of self-attention”. In: *International Conference on Machine Learning*. PMLR, pp. 5562–5571.
Kingma, Diederik P and Max Welling (2013). “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114*.
Kipf, Thomas, Elise van der Pol, and Max Welling (2019). “Contrastive Learning of Structured World Models”. In: *International Conference on Learning Representations*.
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.
Kosiorek, Adam et al. (2019). “Stacked capsule autoencoders”. In: *Advances in neural information processing systems* 32.
Krizhevsky, Alex, Geoffrey Hinton, et al. (2009). “Learning multiple layers of features from tiny images”. In: *Tech Report*.
Kurakin, Alexey, Ian Goodfellow, and Samy Bengio (2016). “Adversarial examples in the physical world”. In: *arXiv preprint arXiv:1607.02533* c, pp. 1–14. arXiv: `1607.02533`.
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, Corinna Cortes, and Christopher JC Burges (1998a). “The MNIST database of handwritten digits, 1998”. In: *URL http://yann.lecun.com/exdb/mnist* 10, p. 34.
LeCun, Yann et al. (1998b). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.
Lee, Juho et al. (2019). “Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks”. In: *International Conference on Machine Learning*, pp. 3744–3753.

Leibig, Christian et al. (Dec. 2017). “Leveraging uncertainty information from deep neural networks for disease detection”. En. In: *Scientific Reports* 7.1, p. 17816. URL: https://doi.org/10.1038/s41598-017-17876-z.

Li, Yingzhen, John Bradshaw, and Yash Sharma (2019). “Are generative classifiers more robust to adversarial attacks?” In: *International Conference on Machine Learning*. PMLR, pp. 3804–3814.

Li, Yingzhen and Yarin Gal (2017). “Dropout inference in Bayesian neural networks with alpha-divergences”. In: *International conference on machine learning*. PMLR, pp. 2052–2061.

Liu, Jeremiah Zhe et al. (2020). “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness”. In: *Advances in Neural Information Processing Systems 33*.

Locatello, Francesco et al. (2020). “Object-centric learning with slot attention”. In: *Advances in Neural Information Processing Systems* 33, pp. 11525–11538.

Louizos, Christos et al. (2017). “Causal effect inference with deep latent-variable models”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6449–6459.

Lyle, Clare et al. (2020). “A bayesian perspective on training speed and model selection”. In: *Advances in Neural Information Processing Systems* 33, pp. 10396–10408.

Mackay, David (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

MacKay, David JC (1992). “A practical Bayesian framework for backpropagation networks”. In: *Neural computation* 4.3, pp. 448–472.

Maddison, C, A Mnih, and Y Teh (2017). “The concrete distribution: A continuous relaxation of discrete random variables”. In: *Proceedings of the international conference on learning Representations*. International Conference on Learning Representations.

Malinin, Andrey and Mark Gales (2018). “Predictive uncertainty estimation via prior networks”. In: *Advances in neural information processing systems* 31.

Mateo-Garcia, Gonzalo et al. (2021). “Towards global flood mapping onboard low cost satellites with machine learning”. In: *Scientific reports* 11.1, pp. 1–12.

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

Meinke, Alexander and Matthias Hein (2019). “Towards neural networks that provably know when they don’t know”. In: *International Conference on Learning Representations*.

Miyato, Takeru et al. (2018). “Spectral Normalization for Generative Adversarial Networks”. In: *International Conference on Learning Representations*.

Mukhoti, Jishnu et al. (2021). “Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty”. In: *arXiv preprint arXiv:2102.11582*.

Nalisnick, Eric et al. (2019a). “Do Deep Generative Models Know What They Don’t Know?” In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1xwNhCcYm.

— (2019b). “Hybrid models with deep and invertible features”. In: *International Conference on Machine Learning*. PMLR, pp. 4723–4732.

Neal, Radford M (1995). “Bayesian Learning for Neural Networks”. PhD Thesis. University of Toronto.

Netzer, Yuval et al. (2011). "Reading digits in natural images with unsupervised feature learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning.*

Nguyen, Anh, Jason Yosinski, and Jeff Clune (2015). "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436.

Ober, Sebastian W, Carl E Rasmussen, and Mark van der Wilk (2021). "The promises and pitfalls of deep kernel learning". In: *Uncertainty in Artificial Intelligence.* PMLR, pp. 1206–1216.

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature visualization". In: *Distill* 2.11, e7.

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748.*

Papernot, Nicolas et al. (2016). "Distillation as a defense to adversarial perturbations against deep neural networks". In: *2016 IEEE symposium on security and privacy (SP).* IEEE, pp. 582–597.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32.*

Peck, Jonathan et al. (2017). "Lower bounds on the robustness to adversarial perturbations". In: *Advances in Neural Information Processing Systems* 30.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12.Oct, pp. 2825–2830.

Pollard, David (2002). *A user's guide to measure theoretic probability.* 8. Cambridge University Press.

Qin, Yao et al. (2019). "Detecting and diagnosing adversarial images with class-conditional capsule reconstructions". In: *arXiv preprint arXiv:1907.02957.*

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

Rasmussen, C.E. and C.K.I. Williams (2006). *Gaussian Processes for Machine Learning.* Adaptative computation and machine learning series. University Press Group Limited. URL: `https://books.google.co.uk/books?id=vWtwQgAACAAJ`.

Rawat, Ambrish, Martin Wistuba, and Maria-Irina Nicolae (2017). "Adversarial Phenomenon in the Eyes of Bayesian Deep Learning". In: *arXiv preprint arXiv:1711.08244.*

Rawlinson, David, Abdelrahman Ahmed, and Gideon Kowadlo (2018). "Sparse unsupervised capsules generalize better". In: *arXiv preprint arXiv:1804.06094.*

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". en. In: *Proceedings of the 31st International Conference on Machine Learning.* Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, pp. 1278–1286. URL: `http://proceedings.mlr.press/v32/rezende14.html`.

Sabour, Sara, Nicholas Frosst, and Geoffrey E Hinton (2017). "Dynamic routing between capsules". In: *Advances in neural information processing systems*, pp. 3856–3866.

Sedghi, Hanie, Vineet Gupta, and Philip M Long (2018). "The Singular Values of Convolutional Layers". In: *International Conference on Learning Representations.*

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.

Shamir, Adi et al. (Jan. 2019). "A simple explanation for the existence of adversarial examples with small hamming distance". In: *arXiv preprint arXiv:1901.10861*. arXiv: `1901.10861`.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Smith, **Lewis** and Yarin Gal (2018). "Understanding measures of uncertainty for adversarial example detection". en. In: *Uncertainty in Artificial Intelligence*, pp. 560–570.

Smith, **Lewis** et al. (2020). "Capsule Networks–A Probabilistic Perspective". In: *arXiv preprint arXiv:2004.03553*.

Smith, **Lewis** et al. (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.

Srivastava, Nitish, Hanlin Goh, and Ruslan Salakhutdinov (2019). "Geometric capsule autoencoders for 3d point clouds". In: *arXiv preprint arXiv:1912.03310*.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of machine learning research* 15.1, pp. 1929–1958.

Storkey, AJ and CKL Williams (2003). "Image modeling with position-encoding dynamic trees". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.7, pp. 859–871.

Sun, Shengyang et al. (2019). "Functional variational bayesian neural networks". In: *arXiv preprint arXiv:1903.05779*.

Sun, Weiwei et al. (2020). "Canonical capsules: Unsupervised capsules in canonical pose". In: *arXiv preprint arXiv:2012.04718*.

Szegedy, Christian et al. (2013). "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199*, pp. 1–10. arXiv: `1312.6199v4`.

Tanay, Thomas and Lewis Griffin (2016). "A boundary tilting persepective on the phenomenon of adversarial examples". In: *arXiv preprint arXiv:1608.07690*.

Tieleman, Tijmen (2014). *Optimizing neural networks that generate images*. University of Toronto (Canada).

— (n.d.). *AffNIST*. `http://www.cs.toronto.edu/~tijmen/affNIST/`. Accessed: 2019-01-09.

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

Turing, Alan (1950). "Computing Machinery And Intelligence". In: *Mind* 59.236, p. 433.

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

Vaswani, Ashish et al. (2017). "Attention Is All You Need". en. In: p. 11.

Wallace, Gregory K (1992). "The JPEG still picture compression standard". In: *IEEE transactions on consumer electronics* 38.1, pp. xviii–xxxiv.

Walmsley, Mike et al. (Oct. 2019). "Galaxy Zoo: Probabilistic Morphology through Bayesian CNNs and Active Learning". In: *Monthly Notices of the Royal Astronomical Society*. stz2816. eprint: `http://oup.prod.sis.lan/mnras/advance-article-pdf/doi/10.1093/mnras/stz2816/30115604/stz2816.pdf`. URL: `https://doi.org/10.1093/mnras/stz2816`.

Weiler, Maurice and Gabriele Cesa (2019). “General $ E (2) $-Equivariant Steerable CNNs”. In: *arXiv preprint arXiv:1911.08251*.
Wilk, Mark van der et al. (2018). “Learning invariances using the marginal likelihood”. In: *Advances in Neural Information Processing Systems* 31.
Wilson, Andrew G et al. (2016a). “Stochastic variational deep kernel learning”. In: *Advances in Neural Information Processing Systems*, pp. 2586–2594.
Wilson, Andrew Gordon et al. (2016b). “Deep kernel learning”. In: *Artificial intelligence and statistics*, pp. 370–378.
Xiao, Han, Kashif Rasul, and Roland Vollgraf (Aug. 2017). “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747*.
Yang, Zichao et al. (2015). “Deep fried convnets”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1476–1483.
Zagoruyko, Sergey and Nikos Komodakis (2016). “Wide Residual Networks”. In: *BMVC*.
Zhang, Richard (2019). “Making convolutional networks shift-invariant again”. In: *International Conference on Machine Learning*. PMLR, pp. 7324–7334.
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.
Zhao, Yongheng et al. (2020). “Quaternion equivariant capsule networks for 3d point clouds”. In: *European Conference on Computer Vision*. Springer, pp. 1–19.

Appendices

A

Supplementary material for Chapter 4

A.1 Deriving the variational bound

In the main body of this chapter, the full derivation of the ELBO was omitted, but the independence assumptions and form of the bound are discussed. Here we provide a full derivation.

We wish to approximate the posterior $p(t^{0:N}_{1:n_k}, A^{0:N}_{1:n_k} \mid X)$. Here the upper index is into layers of random variables, and the lower index is over capsules in that layer. Although we only use two 'hidden layers' in this work, we derive a general bound, as it is little extra effort.

Note, as discussed in the text, we do not apply a variational distribution over s, since the conditional independence assumptions are such that it can be summed out of the variational bound. In more detail, conditional on $A^k_{1:n_k}, t^k_{1:n_k}, A^{k+1}_{1:n_{k+1}}, t^{k+1}_{1:n_{k+1}}$, the activations and poses of the parents, the s^{k+1}_i become conditionally independent of each other.

This will appear in the bound implicitly, as it lets us calculate

$$p(A^{k+1}_i, t^{k+1}_i \mid A^k_{1:n_k}, t^k_{1:n_k}) = \sum_{j \in n_k} p(A^{k+1}_i, t^{k+1}_i \mid A^k_j, t^k_j, s^{k+1}_i = j) p(s^{k+1}_i = j \mid A^k_j, t^k_j) \tag{A.1}$$

Implicitly, this corresponds to an 'E' step in EM; since we will take a gradient step to maximise variational distribution over the ts and As holding the implicit 'soft assignment' over the discrete variables constant.

The variational bound takes the form

$$\mathcal{L}[q] = \mathbb{E}_{q(t^{0:N}_{1:n_k}, A^{0:N}_{1:n_k})} \left[\log p(t^{0:N}_{1:n_k}, A^{0:N}_{1:n_k}, X) - \log q(t^{0:N}_{1:n_k}, A^{0:N}_{1:n_k}) \right]$$

In order to make it clear how this can be computed, we note it can be split up layerwise, using the factorisation of the graphical model that each set of random variables depends only on its immediate parents

$$\begin{aligned}\log p(t^{0:N}_{1:n_k}, A^{0:N}_{1:n_k}, X) = \log p(X \mid t^{N}_{1:n_N}, A^{N}_{1:n_N}) \\ + \log p(t^{N}_{1:n_N}, A^{N}_{1:n_N} \mid t^{N-1}_{1:n_{N-1}}, A^{N-1}_{1:n_{N-1}}) \\ \vdots \\ + \log p(t^{k+1}_{1:n_{k+1}}, A^{k+1}_{1:n_{k+1}} \mid t^{k}_{1:n_k}, A^{k}_{1:n_k}) \\ \vdots \\ + \log p(t^{1}_{1:n_1}, A^{1}_{1:n_1} \mid t^{0}_{1:n_0}, A^{0}_{1:n_0}) \\ + \log p(t^{0}_{1:n_0}, A^{0}_{1:n_0})\end{aligned}$$

We assume a similar layerwise structure for the variational posterior, so that q factorises as

$$\begin{aligned}\log q(t^{0:N}_{1:n_k}, A^{0:N}_{1:n_k}) = \log q(t^{N}_{1:n_N}, A^{N}_{1:n_N} \mid X) \\ + \log q(t^{N-1}_{1:n_{N-1}}, A^{N-1}_{1:n_{N-1}} \mid t^{N}_{1:n_N}, A^{N}_{1:n_N}) \\ \vdots \\ + \log q(t^{k}_{1:n_k}, A^{k}_{1:n_k} \mid t^{k+1}_{1:n_{k+1}}, A^{k+1}_{1:n_{k+1}}) \\ \vdots \\ + \log q(t^{0}_{1:n_0}, A^{0}_{1:n_0} \mid t^{1}_{1:n_1}, A^{1}_{1:n_1})\end{aligned}$$

that is, we assume that the posterior over the random variables at layer k only depends on the variables in the layer immediately below. We also make a mean field assumption within a layer, that is, we assume $q(t^k_{1:n_k}, A^k_{1:n_k} \mid ...) = \prod q(t^k_i \mid ...)q(A^k_i \mid ...)$, but this is not critical for the following derivation and clutters notation, so we omit it from consideration for now.

We can therefore group terms in the variational bound by layer as follows

$$\begin{aligned}\log p(t^{0:N}_{1:n_k}, A^{0:N}_{1:n_k}, X) - \log q(t^{0:N}_{1:n_k}, A^{0:N}_{1:n_k}) = \log p(X \mid t^{N}_{1:n_N}, A^{N}_{1:n_N}) \\ + \log p(t^{N}_{1:n_N}, A^{N}_{1:n_N} \mid t^{N-1}_{1:n_{N-1}}, A^{N-1}_{1:n_{N-1}}) - \log q(t^{N}_{1:n_N}, A^{N}_{1:n_N} \mid X) \\ \vdots \\ + \log p(t^{k}_{1:n_k}, A^{k}_{1:n_k} \mid t^{k-1}_{1:n_{k-1}}, A^{k-1}_{1:n_{k-1}}) - \log q(t^{k}_{1:n_k}, A^{k}_{1:n_k} \mid t^{k+1}_{1:n_{k+1}}, A^{k+1}_{1:n_{k+1}}) \\ \vdots \\ + \log p(t^{0}_{1:n_0}, A^{0}_{1:n_0}) - \log q(t^{0}_{1:n_0}, A^{0}_{1:n_0} \mid t^{1}_{1:n_1}, A^{1}_{1:n_1})\end{aligned}$$

Notice that, apart from the likelihood term $p(X \mid t^{N}_{1:n_N}, A^{N}_{1:n_N})$, these terms all take the form of relative entropies $D_{KL}(p : q) = \mathbb{E}_q[\log p(x) - \log q(x)]$.

A MC estimator of the evidence lower bound could therefore be computed in a layerwise fashion, first sampling from $q(t^{N}_{1:n_N}, A^{N}_{1:n_N} \mid X)$, computing the likelihood term and the KL term for that layer, then computing $q(t^{N-1}_{1:n_N}, A^{N-1}_{1:n_N} \mid X)$, and proceeding layerwise upwards, as only the variables in the layer immediately above or below are ever needed to compute all of these terms.

In practice, as mentioned in the text, we use a bound estimator of this form automatically derived by Pyro's tracing mechanics rather than keeping track of all this bookkeeping manually in our code. This means in practice that we use stochastic estimators of the KL terms even when in principle deterministic expressions could be used, as Pyro does not currently support analytic KL divergences.

A.2 Model implementation details

Here we detail some more mundane details of the model. These could be found by reading the code, to be released, but we record them for greater reproducibility.

The model has a number of hyperparameters, other than the learning rates and neural network hyperparameters and the number of capsules to use in each layer.

The use of Concrete distributions over the t variables introduces a temperature hyperparameter. In addition, the standard deviation parameters c_{ij} and the standard deviation of the pixel noise σ have to be constrained - allowing these to be learned freely allows increasing terms in the variational bound without limit. A more general solution would be to place hyperpriors over these parameters, but for simplicity we reparameterise these so that there is a minimum value each can take, which is a hyperparameter. The minimum c and σ and the temperature are fairly important, and modifying them has a fairly significant effect on performance. It is possible that the training of the model could be improved by, for example, using a temperature schedule, but we did not experiment with this.

The binding probability ρ of the dummy parent and the width λ_{off} of inactive poses are hyperparameters. In our experiments, the model seemed relatively insensitive to these. We set the pixel standard deviation to 0.2. We learn the covariance noise, but set it to have a minimum value of 0.1. We use a temperature of 1.0 for all relaxed variables.

For the convnet that maps from the image to the lowest level latent variables, we use 2 convolutional layers with kernel size 3 and a stride of 2, followed by 2 convolutional layers with kernel size 3 and stride 1 and attentive pooling. We use ELU nonlinearities.

For the set transformer, we use 4 ISAB layers with 128 hidden units, 32 inducing points and 4 attention heads, with layer norm, followed by a PMA block and a linear layer to a 32 dimensional encoding. Independent one layer MLPs with 128 hidden units are used to map from this encoding to the parameters of all variational distributions.

A.3 Details of SCAE Hyperparameters

Kosiorek et al. (2019) report using 24 object and part capsules for MNIST, and setting all the auxiliary losses to 1 except the posterior losses, which are set to 10. However, in the public code release (`https://github.com/google-research/google-research/tree/master/stacked_capsule_autoencoders`), the default script for MNIST uses different hyperparameter settings for these losses than reported in the paper, and 40 object and part capsules. Namely, the `run_mnist` script uses a posterior between example sparsity of 0.2, a posterior within example sparsity weight of 0.7, a prior between example sparsity weight of 0.35, a prior within example constant 4.3 and a a prior within example sparsity weight of 2.

For the SCAE results reported in the paper, we use 24 capsules as in Kosiorek et al. (2019), but we use the loss settings from the code repository, since we tried both and found these to work slightly better. We do not use more capsules because we find this could lead to confusion with the main point of this experiment, which is to test the degree to which the model is capable of separating out pose from visual appearance - having more capsules allows a great deal of redundancy, for example, learning several different objects for a 2 rotated by various degrees, rather than recognising that an image can be explained as a single object in different orientations. As rotated MNIST is identical to normal MNIST apart from the variation in object orientation, it should not in principle require more capsules to explain this data with the generative model.

A.4 SCAE: Prior and Posterior Presences

Kosiorek et al. (2019) report two ways of quantifying the presence of object capsules - the 'prior' presence $a_k \max_m a_{k,m}$ (in the notation of that paper) and the 'posterior' presence $\sum_m a_k a_{k,m} \mathcal{N}(x_m, k, m)$. We report results above with the posterior presence, as in the paper, which performs slightly better. In Kosiorek et al. (2019), these are not explicitly modelled as random variables. The model described in our paper does not correspond exactly to that in Kosiorek et al. (2019), as discussed, but if analogies are drawn, then the a_k correspond to t, and the parameters $a_{k,m}$ correspond to the parent child affinities ρ (we treat these as constant, whereas these are modifiable per input in that work).

The 'prior' probability is roughly equivalent to our use of the logit of the posterior over t_0 for classification. The posterior presence has no immediately obvious correspondence to the posterior over any random variables in our model, but it can be interpreted as the expected number of child capsules that couple to a parent. This can be seen as follows;

The posterior over s conditional on the parent and child poses and presences can be straightforwardly calculated using Bayes rule:

$$p(s_j = i \mid t_j^1, A_j^1, t_{1:n}^0, A_{1:n}^0) \propto p(t_j^1, A_j^1 \mid s_j = i, t_{1:n}^0, A_{1:n}^0) p(s = i \mid t_{1:n}^0, A_{1:n}^0)$$

That is to say, the posterior probability of child j attaching to parent i is given by the above quantity. If we sum this over the children, then we get a very similar expression to the 'posterior presence' score, ignoring the terms that would correspond to the child presence, which is not assigned a likelihood in the model of Kosiorek et al. (2019). This sum over the children is the expected number of children attached to parent i under the posterior distribution.

B Supplementary Material for Chapter 7

B.1 Model details

DUE is an instance of DKL (Wilson et al., 2016b) and uses the sparse GP of Titsias (2009) and the variational approximation of Hensman et al. (2015). In this section we give a complete description of the model.

B.1.1 Full model definition

Let $X \in \mathbb{R}^{N \times D}$ and $Y \in \mathbb{R}^{N \times T}$ be a dataset of N points with input dimensionality D and output dimensionality T. For classification tasks, T is the number of classes, with a single instance $\boldsymbol{y} \in Y$ being a T dimensional vector of class probabilities. Let $F \in \mathbb{R}^{N \times T}$ 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 Y and F, evaluated at inputs X, is

$$p(Y, F; X) = p(Y \mid F) \prod_{t=1}^{T} p(F_{[:,t]}; X) \tag{B.1}$$

$$p(F_{[:,t]}; X) = \mathcal{GP}(\mu_t(X), k_{l_t,\theta}(X, X)). \tag{B.2}$$

Here $F_{[:,t]}$ refers to the tth column vector of the matrix F. $\mu_t(\cdot)$ is a mean function for output dimension t. For both regression and classification we use a constant mean function $\mu_t(X) = \mu_t$, where μ_t is a hyperparameter. $k_{l,\theta}(\cdot, \cdot)$ is a deep kernel function with feature extractor parameters θ, as we discuss in Section 7.1.1, and base kernel $\bar{k}_{l_t}(\cdot, \cdot)$ with hyperparameters l_t specific to each output dimension (but shared for each input dimension). $p(Y \mid F)$ is the likelihood function. For regression tasks this is defined as $p(Y \mid F) = \prod_{i=1}^{N} \mathcal{N}(Y_{[i,:]} \mid F_{[i,:]}, \sigma^2 I_T)$, where σ^2 is a variance hyparameter and I is the identity matrix. For classification tasks,

$$p(Y \mid F) = \prod_{i=1}^{N} p(Y_{[i,:]} \mid F_{[i,:]}) \tag{B.3}$$

$$p(Y_{[i,:]} \mid F_{[i,:]}) = \mathrm{softmax}(F_{[i,:]})_{(\mathrm{argmax}_c\, Y_{[i,c]})}. \tag{B.4}$$

Note that while μ_t, σ^2, and l_t are described as hyperparameters, we do not specify them manually but instead learn them alongside the other model parameters, as below.

B.1.2 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 with the number of data points, thus it is not suitable for large datasets. Thus, for both regression and classification we use a sparse GP approximation and variational inference.

We use the sparse GP approximation of Titsias (2009) which augments the model with M inducing inputs, $Z \in \mathbb{R}^{M \times J}$, where J is the dimensionality of the feature space. The associated inducing variables, $U \in \mathbb{R}^{M \times T}$, 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 Z are treated as variational parameters. U are random variables with prior $p(U) = \prod_{t=1}^{T} \mathcal{N}(U_{[:,t]} | \mu_t(Z), \bar{k}(Z, Z))$, and variational posterior $q(U) = \prod_{t=1}^{T} \mathcal{N}(U_{[:,t]} | \boldsymbol{m}_t, S_t)$, where $\boldsymbol{m}_t \in \mathbb{R}^M$ and $S_t \in \mathbb{R}^{M \times M}$ are variational parameters and initialised at the zero vector and the identity matrix respectively. The approximate predictive posterior distribution at test points X^* is then

$$q(F^* \mid Y; X, X^*) = \int p(Y \mid F^*) p(F^* \mid U; X^*, Z) \prod_{t=1}^{T} q(U_{[:,t]} \mid \boldsymbol{m}_t, S_t) \, \mathrm{d}U \, \mathrm{d}F^*. \quad \text{(B.5)}$$

Here $p(F^* \mid U; X^*, Z)$ 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 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 Z, $\boldsymbol{m}_t$, and S_t, alongside the feature extractor parameters θ and model hyperparameters μ_t, l_t, and σ^2, are all learned by maximising a lower bound on the log marginal likelihood, known as the ELBO, $\mathcal{L}$. For the variational approximation above, this is defined as

$$p(Y; X) \geq \mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_{q(F_{[i,:]} \mid \boldsymbol{m}_1, \ldots, \boldsymbol{m}_T, S_1, \ldots, S_T; \boldsymbol{x}_i, Z)} \left[\log p(Y_{[i,:]} \mid F_{[i,:]}) \right] - D_{\mathrm{KL}}(q(U) || p(U)). \quad \text{(B.6)}$$

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 and hyperparameters, and variational parameters, using stochastic gradient descent. To accelerate optimisation we additionally use the whitening procedure of Matthews (2017). We specify the precise optimiser configuration for each experiment later in this section.

B.1.3 Making predictions and measuring uncertainty

For regression tasks we directly use the function values F^* above as the predictions. We use the mean of $q(F^* \mid Y; X, X^*)$ as the prediction, and the variance as the uncertainty.

For classification tasks we need the posterior over the class probabilities, $q(Y^*|Y;X,X^*)$, rather than the latent function values. Thus, we approximate the integral

$$\bar{Y}^* = \int \mathrm{softmax}(F^*) q(F^* \mid Y; X, X^*) \, \mathrm{d}F^*, \tag{B.7}$$

using Monte Carlo samples (32 in practice), which are very fast to compute and do not require additional forward passes. Note that we consider the inputs *independent* at test time (i.e. we only take the diagonal of the posterior covariance) which is especially important when detecting out of distribution data.

The predicted class for input i is then the most likely class in $\bar{Y}^*_{[i,:]}$. To estimate the uncertainty of the prediction we compute the entropy of $\bar{Y}^*_{[i,:]}$:

$$\mathrm{entropy}(\bar{Y}^*_{[i,:]}) = -\sum_c \bar{Y}^*_{[i,c]} \log \bar{Y}^*_{[i,c]}. \tag{B.8}$$

B.1.4 Implementation

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 of the 1,000 points. The models are implemented using GPyTorch (Gardner et al., 2018) and PyTorch (Paszke et al., 2019) and use their default values if not otherwise specified.

B.2 Experimental Details

B.2.1 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 Lipschitz constant of 0.65, five power iterations, an RBF kernel and additive Gaussian noise with $\sigma = 0.1$ for both. For toy regression, we use 50 inducing points and for two moons we use four.

B.2.2 CIFAR-10

For the WRN, we follow the experimental setup and implementation of Zagoruyko and Komodakis (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. We use momentum 0.9 and weight decay 5e-4. We select 20% of the training data at random as a validation set for hyper-parameter tuning, but obtain the final model by using the full training set with the final set of hyper-parameters. SV-DKL was trained

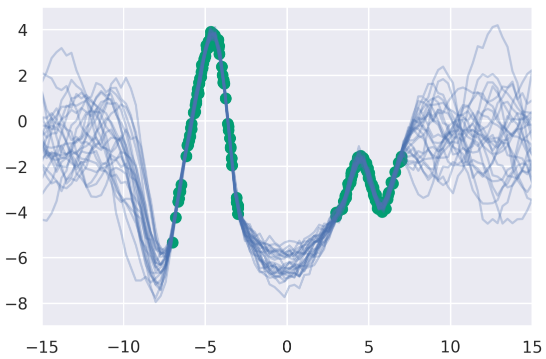


Figure B.1: A visualisation of draws from the posterior using the Matern kernel with $\nu = \frac{1}{2}$

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., 2021). We use 1 power iteration and use the lowest Lipschitz constant 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.

In the out-of-distribution detection experiment, it is crucial to preprocess the SVHN images in the same way as CIFAR-10, as we cannot know at test time from which dataset the image comes and the only sensible procedure is to preprocess as if it was coming from the training dataset.

B.2.3 Regression Experiment

Following Shalit et al. (2017), we use 63%/27%/10% train / validation / test splits and report the RMSE 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.

For the DKLITE experiments we use the open source code from the authors[1]. We write a custom loop over the IHDP dataset to follow the above protocol.

[1] `https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/master/alg/dklite/`

C

Supplementary Material for Chapter 8

C.1 Background: The Nyquist-Shannon Theorem And Aliasing

In this section, we give a more detailed discussion of the Nyquist-Shannon theorem and aliasing, and also treat some of the subtleties of this result in more detail than was possible in the main body.

The classical application of the Nyquist-Shannon theorem is the digitisation of a continuous signal; if a signal sampled at a frequency u_s contains no frequency content above the Nyquist frequency $u_s/2$ then it can be reconstructed exactly. However, the result applies equally to re-sampling at a lower frequency.

It is worth mentioning that the Nyquist-Shannon theorem is a *sufficient* condition for lossless reproduction, not a necessary one; with some additional assumptions it is possible to resolve an 'undersampled' signal. The most notable application of this is in compressed sensing, where a signal can sometimes be recovered exactly if it is known that it is sparse in the spectral domain.

In the main body, we refer to aliasing and to low-passing the image to prevent this phenomenon. An illustration of the effect on aliasing on the frequency spectrum of a (1D) signal is shown in Figure C.1. An anti-aliasing filter is one like that in the second row of this figure; we remove all frequency above the Nyquist rate to reduce any aliasing effects. Of course, this still causes feature collapse (indistinguishable signals), but the resultant feature collapse is far easier to reason about, as well as being better behaved; uncontrolled aliasing introduces non-local artefacts into a signal, which in fact hurts the translation invariance of convolutional classifiers (Zhang, 2019). A simple dimensionality argument shows that the volume of potential signals made indistinguishable by these two operations is the same. Say we have a linear mapping D from a signal in $\mathbb{R}^n$ to one in $\mathbb{R}^{n/2}$. The rank-nullity theorem implies that each signal $y \in \mathbb{R}^{n/2}$ has a space of dimension $\mathbb{R}^{n/2}$ of solutions x to $y = Dx$. This applies whether D is naive decimation or anti-alias filtering + naive decimation, as both of these are linear operations.

In this chapter, we often assume 'ideal' low pass filtering, that is, a filter which lets all content below a given frequency pass through unchanged, while setting all high frequency content to zero. This is possible for discrete signals (by implementing this as a direct binary mask in the Fourier domain), but is rarely

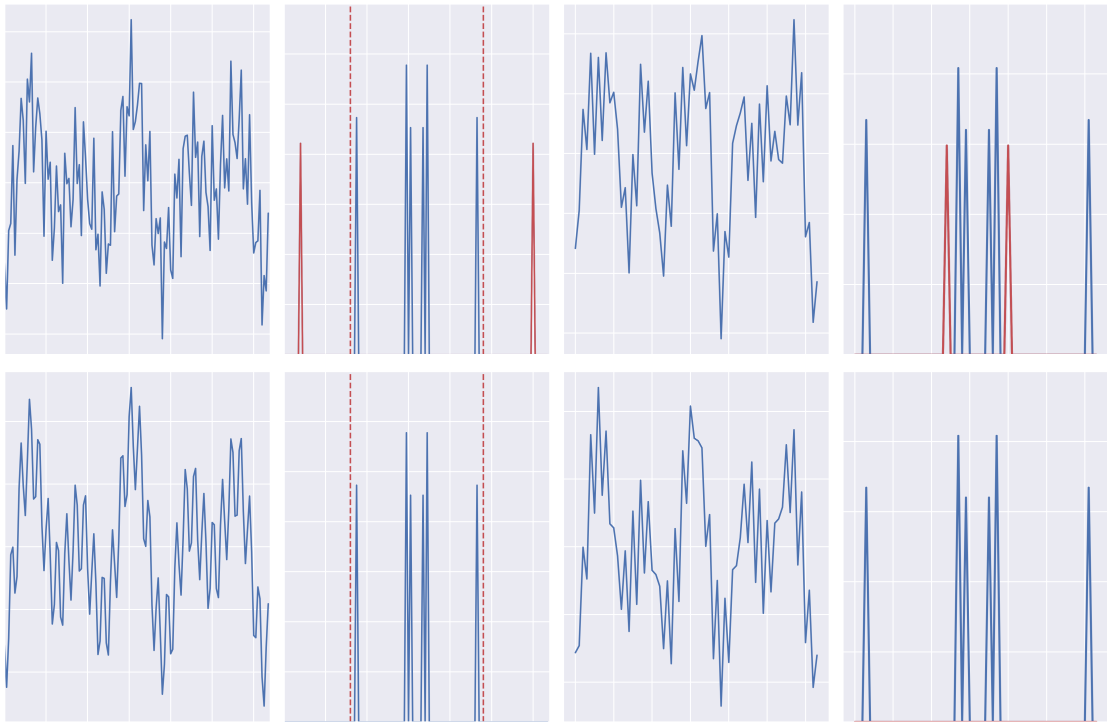


Figure C.1: Figure illustrating aliasing in a 1D signal after decimation by a factor of 2. From left to right; a) the original signal. b) the Fourier spectrum of the original signal, with the Nyquist frequency after downsampling marked as a red dotted line and content above the Nyquist marked in red. Frequency content above this limit will be aliased after downsampling. c). the decimated signal. d) the Fourier spectrum of the decimated signal. The peak due to the aliasing of the high frequency component of the original signal is marked in red. The second row shows the same, but with a low-pass filter applied to the original signal to remove content above the Nyquist limit before downsampling. The frequency content above the Nyquist is simply lost, but the sub-Nyquist content passes through undistorted.

done. For continuous signals this ideal frequency response cannot be realised. This kind of filtering is often considered undesirable in signal processing because a filter with bounded support in the frequency domain, like the brick wall filter, must have unbounded support in the spatial domain, due to the uncertainty principle for Fourier transforms. As the Fourier transform assumes that a signal is periodic forever outside the bounds of the image, this means that this filter mixes an image spatially with the repeated copies of itself outside the bounds of the image due to the unbounded support of the filter, leading to characteristic 'ringing' artefacts. In addition, implementing it for discrete signals requires a Fourier transform, rather than using the direct convolution; although the Fourier transform implementation actually has lower asymptotic complexity than a direct convolution, in practice the direct operation is highly optimised on modern GPUs and can often be faster. Zhang (2019)'s method BlurPool uses a direct convolution kernel to implement this anti-aliasing filter step before all downsampling operations, which uses a 'soft' low pass filter with bounded spatial support rather than a brick wall.

In the text, we claim that the power of typical images is concentrated in the low frequencies. In Figure 8.4 we show the spectra of a random sample of images, showing that indeed most have their power concentrated in the low frequencies, though we do not find that they are universally low frequency dominant in the sense of Lemma C.3, as shown in Tables C.1 and C.2. However, this does motivate why this assumption of 'low frequency dominance' is not unreasonable for natural images. In addition, this does suggest that even though the low frequency interval may not *strictly* be dominant in the sense we use to prove Lemma C.3, as we point out in Section 8.4 this is likely to be a weak sufficient condition, and if images power is broadly concentrated in the low frequencies it seems reasonable to expect that the ReLU (since it is a convolution in the frequency domain) will still reduce the distances in the low frequency band, which could be an explanation for why we see low-frequency contraction holding far more often than our sufficient condition.

C.2 Proofs Omitted From The Main Text

C.2.1 Proofs for Section 8.2.2

Proof of Theorem 8.2. As mentioned in the main text, the fact that a mapping from $\mathbb{R}^n \to \mathbb{R}^m$ with $m \neq n$ cannot be bilipschitz follows from the fact that a bilipschitz mapping is necessarily a homeomorphism, and the fact that $\mathbb{R}^n$ and $\mathbb{R}^m$ are not homeomorphic to each other, which is a consequence of the invariance of domain theorem (Brouwer, 1911). We proceed to establish these claims: First, we prove the following lemma.

Lemma C.1. *Let $f : X \mapsto Y$ be a bilipschitz function, so $\frac{1}{L}||x_1 - x_2||_X \leq ||f(x_1) - f(x_2)||_Y \leq L||x_1 - x_2||_X$, and let $Y = f(X)$ be the image of X under f. Then f is a homeomorphism between X and Y, and so X and Y are homeomorphic.*

Proof. Recall that a function f is a homeomorphism if f is bijective, f is continuous, and f^{-1} is also continuous. We will address these in turn. To see that f is bijective, note that f is injective iff. $\forall x_1, x_2 \in X$, we have $x_1 \neq x_2 \implies f(x_1) \neq f(x_2)$. But this follows directly from the lower Lipschitz property of f, since if $x_1 \neq x_2$, then $||x_1 - x_2||_X > 0$, so $||f(x_1) - f(x_2)||_Y > 0$, from which it follows that $f(x_1) \neq f(x_2)$. Since f is injective (one-to-one) and surjective (since $Y = f(X)$), it is a bijection. Since any function which is Lipschitz continuous is also continuous, the fact that f is continuous is given. Since f is bijective, the inverse function f^{-1} exists, and we need to show that it is continuous. We have, from the bilipschitzness of f, that $\frac{1}{L}||f^{-1}(f(x_1)) - f^{-1}(f(x_2))||_X \leq ||f(x_1) - f(x_2)||_Y$, which implies that the inverse function is also Lipschitz, and hence also continuous. So f is a homeomorphism. $\square$

This proof makes the additional assumption that the function is surjective. If the function is not surjective, then it is a bilipschitz *embedding*, not a bilipschitz function (Druţu and Kapovich, 2018). This is mostly a technical point about the definition of the inverse function which doesn't affect the overall argument much. For instance, consider a linear bilipschitz map $A : \mathbb{R}^n \to \mathbb{R}^m$ with $m > n$. The image of this function will be a linear subspace of dimension n in $\mathbb{R}^m$. This function would be a homeomorphism between $\mathbb{R}^n$ and this subspace, but (obviously) not a homeomorphism between $\mathbb{R}^n$ and $\mathbb{R}^m$, as the inverse would not be well defined outside the image of $A\mathbb{R}^n$.

We also establish the following lemma, which is a well known corollary of the invariance of domain theorem we prove for completeness:

Lemma C.2. *If $n > m$, and U is a non-empty open subset of $\mathbb{R}^n$, there is no injective, continuous mapping between U and $\mathbb{R}^m$. In particular, therefore, $\mathbb{R}^n$ and $\mathbb{R}^m$ are not homeomorphic.*

Proof. Recall the invariance of domain;

Theorem C.1 (Brouwer (1911))**.** *Let U be an open subset of $\mathbb{R}^n$, and let $f : U \to \mathbb{R}^n$ be an injective continuous map. Then $V = f(U)$ is also open in $\mathbb{R}^n$.*

which we will take as given. Suppose an injective, continuous function $f : U \to \mathbb{R}^m$ existed. It follows, then, that we could also consider an extension of f as mapping from $\mathbb{R}^n \to \mathbb{R}^n$, by composing with a function of the form $[x_1, x_2, x_3, \ldots x_m] \in \mathbb{R}^m \to [x_1, x_2, \ldots x_m, 0, 0, \ldots 0] \in \mathbb{R}^n$, for example. f would then map from U to the hyperplane $\mathbb{R}^m \subset \mathbb{R}^n$. But a hyperplane is not an open set; for any point x in the hyperplane there is a point $x + \epsilon$ that is *not* in the hyperplane, where we can make ϵ arbitrarily small. Using Theorem C.1, however, we know that if f is continuous and injective, then its image must be an open set. We therefore conclude by contradiction that $f : U \to \mathbb{R}^m$ cannot be a continuous, injective mapping, and so cannot be a homeomorphism.

□

This proves Theorem 8.2 by contradiction; if a function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ existed and was bilipschitz, then we would have shown that $\mathbb{R}^n$ and $\mathbb{R}^m$ were homeomorphic, using Lemma C.1. But Lemma C.2 shows that it is impossible for such a homeomorphism to exist.

□

A less formal, but more constructive, argument that this is impossible is given in Section 8.6.2.

C.2.2 Proofs for Section 8.3.1

Proof of Lemma 8.1 . By assumption, our ResNet is of the form $f(x) = A(x)+g(x)$, where A and g are both of the form $g = g_1 \circ g_2 \circ \ldots g_n$ where all the component functions g_i are pointwise nonlinearities, dimension preserving convolutions, strided convolutions or normalisation layers like BatchNorm. Firstly, using the result we already have that strided convolutions are equivalent to normal convolutions, plus a decimation operation, we can instead consider g as consisting of nonlinearities, dimension preserving convolutions, decimation operations and BatchNorm. We aim to show we can re-write this as $f(x) = D(A(x) + g(x))$, where D is a standard decimation operation. To do this, we need to show that, we can move all decimation operations to the 'outside' of g and combine them all into a single operation.

D commutes trivially with pointwise operations, which covers nonlinearities and batch-norm; since a pointwise operation acts independently on every pixel in an image, and D drops pixels, it makes no difference whether the decimation is applied before or after the pointwise operation. D can also be combined easily with other decimation operations, since decimation by a factor a followed by decimation by a factor b is exactly the same as a single decimation by ab. Finally, we can also swap the order of a convolution C and decimation by dilating the convolution kernel by an appropriate factor; that is, $C(D(x, a), 1) = D(C(x, a), a)$ where $C(x, a)$ applies a convolution with dilation a, and $D(x, a)$ is decimation with a factor a. Applying all these sequentially, we readily see that we can always move D outside g or A, and it 'factors' out due to the linearity of decimation (i.e $D(A(x)) + D(g(x)) = D(A(x) + g(x))$.

□

C.2.3 Proofs for Section 8.4

Proof of Lemma 8.2. We break this proof down into several lemmas. First, we establish the following lemma about convolution, which uses our domination assumption. For concreteness, we prove this in 2 dimensions, as this is the specific case used in the argument, though the proof can be easily be adapted to be more general.

Lemma C.3. *Let $f(x_1, x_2)$ and $g(x_1, x_2)$ be positive discrete signals on a 2D $N \times N$ grid, and further assume they are both normalised, so $\sum_{x_1} \sum_{x_2} f(x_1, x_2) = 1$ and likewise for g. We assume circular padding, so that $f(x_1, x_2) = f(x_1 \bmod N, x_2 \bmod N)$ if the x_i are outside the range $0, N-1$, but omit this to avoid cluttering notation. Let $f * g = (f * g)(x_1, x_2) = \sum_{t_1=0}^{N-1} \sum_{t_2=0}^{N-1} f(t_1, t_2) g(x_1 - t_1, x_2 - t_2)$ denote the convolution of the functions f and g. Let an interval $[x, x+L] = [x_1, x_1 + L_1] \times [x_2, x_2 + L_2]$ starting at x be a dominant interval in f. Then the mass in that dominant interval is reduced (or stays the same) when f is convolved with g, whatever g is; that is $\sum_{t_1=x_1}^{x_1+L_1} \sum_{t_2=x_2} [f * g](t_1, t_2) \leq C$.*

Proof of Lemma C.3 . The proof of this is fairly straightforward, given the assumptions. These are that f, g are normalised measures, and there exists a dominant interval $[x, x+L]$ in f such that $C = \sum_x^{x+L} f(x) \geq \sum_y^{y+L} \forall y$, abusing notation so $x = x_1, x_2$ The mass of $f * g$ in the interval is (as convolution is symmetric)

$$\begin{aligned}\sum_x^{x+L} [f * g](x) dx &= \sum_{y_1=x_1}^{x_1+L_1} \sum_{y_2=x_2}^{x_2+L_2} \sum_{t_1=0}^{N-1} \sum_{t_2=0}^{N-1} g(t_1, t_2) f(y_1 - t_1, y_2 - t_2) \\ &= \sum_{t_1=0}^{N-1} \sum_{t_2=0}^{N-1} g(t_1, t_2) \sum_{y_1=x_1}^{x_1+L_1} \sum_{y_2=x_2}^{x_2+L_2} f(y_1 - t_1, y_2 - t_2)\end{aligned}$$

However, by our concentration assumption, $\sum_{y_2=x_2}^{x_2+L_2} f(y_1 - t_1, y_2 - t_2) \leq C$. Therefore,

$$\begin{aligned}\sum_x^{x+L} [f * g](x) &\leq \sum_{t_1=0}^{N-1} \sum_{t_2=0}^{N-1} g(t_1, t_2) C \\ &\leq C\end{aligned}$$

due to the normalisation of g. □

We now prove a special case of our main result, which shows the essential idea behind the proof, then introduce another lemma which can be used to extend it to the general case.

Lemma C.4. *Let x be an image, and assume that x is dominated by frequency content below a frequency u (that is, for a 2D signal, the interval $[0, u] \times [0, u]$ is dominant in the sense of definition 8.1). Then the ReLU is a contraction on the low-frequency component of x, that is, $||H_u(\text{ReLU}(x))|| \leq ||H_u(x)||$.*

Proof. Let $z = \text{ReLU}(x)$, and let Z and X be the Fourier transforms of their respective variables. Denote by $[A*]$ the Toeplitz or block-Toeplitz matrix that implements the (discrete, 2D) convolution $A * X$ as matrix multiplication: that is $A * X = [A*]X$. Consider splitting these up into their high frequency and low frequency components, so $Z_l = H_u(Z)$, $Z_h = Z - Z_l$ and likewise for X. Note that these vectors are orthogonal and $Z = Z_l + Z_h$. Recall that we can write the ReLU as a convolution in the frequency domain, $Z = M(x) * X$. The vector norm is invariant to taking absolute values elementwise, so we can consider the norm of $|Z| = |M(x)| * |X|$, which allows these vectors to be treated as unnormalised measures. The fact that we can take the magnitude inside the convolution simply follows from the definition of convolution.

We assumed that X is low-band dominant, as in the conditions of Lemma C.3, so X_l is a dominant interval. It follows then that, applying Lemma C.3,

$$\frac{||Z_l||}{||Z||} \leq \frac{||X_l||}{||X||}$$
$$||Z_l|| \leq ||X_l|| \frac{||Z||}{||X||}$$

Now, recall that $Z = M(x) * X$ is just an application of the ReLU. But the ReLU has a Lipschitz constant of 1, so the operator norm of the matrix $[M(x)*]$ must be less than 1. Therefore, $\frac{||Z||}{||X||} \leq 1$, and we can substitute this into the above to obtain the desired result that

$$||Z_l|| \leq ||X_l||$$

□

This proves that the ReLU will contract the low-frequency norm of an individual input if it is low-frequency dominant, but not between that it will contract the distance between *pairs* of points. We can extend the above proof to this general case using the following lemma;

Lemma C.5. *Let x and $x + v$ be two arbitrary points. Let ϕ be an arbitrary piecewise linear pointwise operator that can be written as $\phi(x) = M(x) \cdot x$ where $\nabla_x M(x) = 0$ (such as the ReLU). There exists $\lambda^\star \in [0, 1]$ such that the following bound holds on the distance between these two points:*

$$||\phi(x + v) - \phi(x)|| \leq ||m(x + \lambda^\star v) \cdot v||$$

or equivalently in frequency space

$$||\phi(X + V) - \phi(X)|| \leq ||M(x + \lambda^\star v) * V||$$

Proof of Lemma C.5 . First note that, using the fundamental theorem of calculus, we have that, for a general differentiable vector valued function $g(x)$ with Jacobian matrix $J(x)$,

$$g(x + v) - g(x) = \int_0^1 J(x + \lambda v) v d\lambda$$

for any x and v. We can take norms on both sides, then apply Jensen's inequality to obtain

$$\begin{aligned} ||g(x + v) - g(x)|| &= || \int_0^1 J(x + \lambda v) v d\lambda|| \\ &\leq \int_0^1 ||J(x + \lambda v) v|| d\lambda \end{aligned}$$

Now, along this line parameterised by λ there must exist $\lambda^\star := \arg\max_\lambda ||J(x + \lambda v)v||$ which maximises this inner product. We can upper bound the norm on the left hand side by replacing the integrand with this value to obtain

$$\begin{aligned}
||g(x+v)-g(x)|| &\leq \int_0^1 ||J(x+\lambda v)v|| d\lambda \\
&\leq \int_0^1 ||J(x+\lambda^\star v)v|| d\lambda \\
&\leq ||J(x+\lambda^\star v)v|| \int_0^1 d\lambda \\
&\leq ||J(x+\lambda^\star v)v||
\end{aligned}$$

But for $\phi(x)$, we have $\phi(x) = m(x) \cdot x$, or $\phi(X) = M(x) * X$ in frequency space, which we are free to consider as this simply represents a change of basis. Therefore, $\nabla_x \phi(X) = [M(x)*]$, and so $||J(x+\lambda^\star v)v|| = ||M(x+\lambda^\star) * V||$

□

So we can upper bound the distance in output space with the norm of a convolution at a single 'worst case' point $x + \lambda^\star$. This 'worst case' mask is at a point along the linear trajectory connecting x and $x + v$. Lemma C.5 can then be used to extend Lemma C.4 to the general case; first, we can upper bound $||\operatorname{ReLU}(X+V) - \operatorname{ReLU}(X)||$ with $||M(x+\lambda^\star v) * V||$. We can then apply the proof of Lemma C.4 to $Z = ||M(x+\lambda^\star v) * V||$, splitting V and Z into low and high frequency components and using the same argument to show that $Z_l \leq V_l$ assuming that V_l is low-frequency dominant, which establishes the claim in Lemma 8.2

□

Proof of Theorem 8.3. Despite the fact that this theorem is a more important result, its proof is essentially a corollary of Lemma 8.2. First, note that if this lemma applies, the ReLU reduces the low-frequency distances, so $||H_u(\operatorname{ReLU}(x)) - H_u(\operatorname{ReLU}(y))|| \leq ||H_u(x) - H_u(y)||$. Convolutions, if we apply spectral normalisation so that the we have $\operatorname{Lip}(g_i) < 1$ for all convolutions g_i in the branch, must also reduce the low-frequency distances (since they act on all components independently), and so we must also have $||H_u(g(x)) - H_u(g(y))|| \leq L||H_u(x) - H_u(y)||$ for some $L < 1$, since g is a composition of functions for which this is true. Using an argument similar to that in (Behrmann et al., 2019), we then can use the residual structure to obtain (assuming that the skip connection is average pooling)

$$\begin{aligned}
&||H_u(f(x)) - H_u(f(y))|| \\
&\qquad = ||H_u(x+g(x)) - H_u(y+g(y))|| \\
&\qquad = ||H_u(x) - H_u(y) - (H_u(g(x)) - H_u(g(y)))|| \\
&\qquad \geq ||H_u(x) - H_u(y)|| - ||H_u(g(x)) - H_u(g(y))|| \\
&\qquad \geq (1-L)||H_u(x) - H_u(y)||
\end{aligned}$$

where the third line uses the reverse triangle inequality, and the fourth line applies Lemma 8.2. □

C.3 Additional Experimental Results

C.3.1 Additional verification results

In the main body of this chapter, we provide the results of checking the conditions of Lemma C.3 and Theorem 8.3 on Fashion MNIST. Here, in Table C.1 and Table C.2, we provide analogous results for CIFAR10 and MNIST in addition. These were omitted from the main text in order to avoid breaking up the flow of the text with too many tables and figures, as the results are consistent across the three datasets tested.

	Dataset	Lip(g)	x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Lemma C.3	MNIST	0.9	0.80 ± 0.00	0.18 ± 0.04	0.31 ± 0.03	0.63 ± 0.02	0.44 ± 0.02
		0.95	0.80 ± 0.00	0.18 ± 0.04	0.30 ± 0.05	0.62 ± 0.04	0.45 ± 0.02
		0.99	0.80 ± 0.00	0.20 ± 0.03	0.31 ± 0.03	0.63 ± 0.02	0.45 ± 0.01
		3.0	0.80 ± 0.00	0.23 ± 0.02	0.43 ± 0.04	0.71 ± 0.01	0.47 ± 0.02
		6.0	0.80 ± 0.00	0.22 ± 0.03	0.42 ± 0.03	0.71 ± 0.01	0.48 ± 0.02
		9.0	0.80 ± 0.00	0.22 ± 0.04	0.41 ± 0.03	0.71 ± 0.01	0.48 ± 0.01
		no	0.80 ± 0.00	0.24 ± 0.02	0.39 ± 0.05	0.70 ± 0.01	0.46 ± 0.02
	FMNIST	0.9	0.86 ± 0.00	0.18 ± 0.04	0.24 ± 0.04	0.48 ± 0.04	0.47 ± 0.03
		0.95	0.86 ± 0.00	0.18 ± 0.04	0.25 ± 0.02	0.48 ± 0.03	0.46 ± 0.01
		0.99	0.86 ± 0.00	0.18 ± 0.05	0.25 ± 0.04	0.51 ± 0.02	0.48 ± 0.01
		3.0	0.86 ± 0.00	0.24 ± 0.03	0.28 ± 0.04	0.53 ± 0.04	0.47 ± 0.02
		6.0	0.86 ± 0.00	0.24 ± 0.03	0.28 ± 0.02	0.53 ± 0.02	0.47 ± 0.02
		9.0	0.86 ± 0.00	0.25 ± 0.01	0.28 ± 0.04	0.54 ± 0.04	0.47 ± 0.01
		no	0.86 ± 0.00	0.24 ± 0.03	0.25 ± 0.03	0.53 ± 0.03	0.47 ± 0.01
Theorem 8.3	MNIST	0.9	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		0.95	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		0.99	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		3.0	n/a	0.69 ± 0.31	0.96 ± 0.11	1.00 ± 0.00	1.00 ± 0.00
		6.0	n/a	0.55 ± 0.26	0.98 ± 0.05	1.00 ± 0.00	1.00 ± 0.00
		9.0	n/a	0.61 ± 0.28	0.99 ± 0.03	1.00 ± 0.00	1.00 ± 0.00
		no	n/a	0.60 ± 0.30	0.99 ± 0.02	1.00 ± 0.00	1.00 ± 0.00
	FMNIST	0.9	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		0.95	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		0.99	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		3.0	n/a	0.98 ± 0.04	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		6.0	n/a	0.95 ± 0.11	0.92 ± 0.04	1.00 ± 0.00	1.00 ± 0.00
		9.0	n/a	1.00 ± 0.01	0.85 ± 0.12	1.00 ± 0.00	1.00 ± 0.00
		no	n/a	1.00 ± 0.00	0.86 ± 0.07	1.00 ± 0.00	1.00 ± 0.00

Table C.1: Results of empirically checking the conditions of Lemma C.3 (low-frequency domination) and Theorem 8.3 (low frequency contraction). Numbers are the proportion of the dataset for which we found the condition to hold exactly; so 1 means no violations were found, 0.5 means it was true for half the inputs tested, etc. We report means and standard error over 25 seeds for these datasets. We use a WideResNet with depth 10 and widen factor of 1 for these datasets.

	Lip(g)	x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
Lemma 1	0.9	0.66 ± 0.00	0.53 ± 0.11	0.56 ± 0.20	0.41 ± 0.13	0.41 ± 0.13	0.41 ± 0.12	0.67 ± 0.01
	0.95	0.66 ± 0.00	0.53 ± 0.03	0.41 ± 0.19	0.35 ± 0.07	0.37 ± 0.06	0.39 ± 0.04	0.67 ± 0.06
	0.99	0.66 ± 0.00	0.52 ± 0.03	0.35 ± 0.17	0.30 ± 0.05	0.30 ± 0.04	0.33 ± 0.04	0.65 ± 0.01
	3.0	0.66 ± 0.00	0.54 ± 0.03	0.26 ± 0.02	0.19 ± 0.03	0.20 ± 0.02	0.26 ± 0.02	0.64 ± 0.01
	6.0	0.66 ± 0.00	0.94 ± 0.00	0.36 ± 0.03	0.28 ± 0.09	0.21 ± 0.06	0.25 ± 0.03	0.62 ± 0.02
	9.0	0.66 ± 0.00	0.88 ± 0.13	0.43 ± 0.04	0.27 ± 0.05	0.20 ± 0.04	0.18 ± 0.02	0.60 ± 0.01
	no	0.66 ± 0.00	0.92 ± 0.03	0.41 ± 0.04	0.24 ± 0.07	0.17 ± 0.02	0.18 ± 0.02	0.60 ± 0.01
	Lip(g)	$f_7(x)$	$f_8(x)$	$f_9(x)$	$f_{10}(x)$	$f_{11}(x)$	$f_{12}(x)$	$f_{13}(x)$
	0.9	0.69 ± 0.03	0.71 ± 0.03	0.78 ± 0.04	0.85 ± 0.08	0.77 ± 0.01	0.78 ± 0.03	0.94 ± 0.01
	0.95	0.67 ± 0.03	0.69 ± 0.02	0.77 ± 0.05	0.81 ± 0.09	0.76 ± 0.02	0.76 ± 0.03	0.95 ± 0.02
	0.99	0.67 ± 0.01	0.69 ± 0.01	0.75 ± 0.04	0.81 ± 0.08	0.77 ± 0.01	0.78 ± 0.02	0.93 ± 0.02
	3.0	0.63 ± 0.02	0.66 ± 0.02	0.72 ± 0.01	0.76 ± 0.02	0.84 ± 0.02	0.81 ± 0.04	0.83 ± 0.03
	6.0	0.63 ± 0.01	0.65 ± 0.01	0.73 ± 0.02	0.85 ± 0.07	0.77 ± 0.01	0.76 ± 0.05	0.82 ± 0.04
	9.0	0.63 ± 0.01	0.66 ± 0.02	0.69 ± 0.01	0.69 ± 0.06	0.76 ± 0.01	0.71 ± 0.05	0.85 ± 0.02
	no	0.63 ± 0.01	0.68 ± 0.02	0.70 ± 0.01	0.75 ± 0.10	0.74 ± 0.03	0.81 ± 0.01	0.74 ± 0.02
Theorem 1	Lip(g)	x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
	0.9	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.95	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.99	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	3.0	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	6.0	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	9.0	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	no	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	Lip(g)	$f_7(x)$	$f_8(x)$	$f_9(x)$	$f_{10}(x)$	$f_{11}(x)$	$f_{12}(x)$	$f_{13}(x)$
	0.9	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.95	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.99	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	3.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	6.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	9.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	no	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00

Table C.2: Verifying theories on CIFAR10 - see the caption of Table C.1 for a more detailed description. We report means and standard errors over 5 seeds for this dataset. We use a WideResNet with a depth of 28 and widen factor of 10.

C.3.2 AUROC and test accuracy

In Table C.4, we show the accuracy and OOD performance, as measured by AUROC on an out-of distribution dataset, in Table C.3. These model do not obtain SotA accuracy, but we include them to demonstrate that we achieve competitive accuracy, in order to demonstrate that our training setup is reasonable and our architectural changes do not have a major effect on the performance of our models.

Ind dataset	OoD dataset	Lip(g) = 0.9	Lip(g) = 0.95	Lip(g) = 0.99	Lip(g) = 3.0	Lip(g) = 6.0	Lip(g) = 9.0	No spec norm
MNIST	FMNIST	99.98 ± 0.01	99.98 ± 0.01	99.98 ± 0.01	99.97 ± 0.01	99.97 ± 0.01	99.97 ± 0.01	99.97 ± 0.01
FMNIST	MNIST	98.75 ± 0.24	98.50 ± 0.47	98.67 ± 0.35	97.89 ± 1.21	98.25 ± 0.52	98.10 ± 0.96	98.28 ± 0.46
CIFAR10	SVHN	93.47 ± 2.81	91.66 ± 1.12	93.27 ± 2.63	94.07 ± 0.70	92.97 ± 2.47	92.31 ± 2.17	95.64 ± 1.58
CIFAR10	CIFAR100	86.92 ± 0.84	86.75 ± 0.64	87.28 ± 0.64	89.74 ± 0.55	89.22 ± 0.52	89.72 ± 0.29	90.16 ± 0.56

Table C.3: OOD detection results (AUROC).

Lip(g)	MNIST	FMNIST	CIFAR10
0.9	99.49 ± 0.05	90.64 ± 0.35	94.54 ± 0.13
0.95	99.55 ± 0.08	90.74 ± 0.30	94.64 ± 0.20
0.99	99.54 ± 0.07	90.97 ± 0.22	94.85 ± 0.24
3.0	99.58 ± 0.08	92.46 ± 0.22	95.75 ± 0.08
6.0	99.56 ± 0.10	92.97 ± 2.47	95.54 ± 0.13
9.0	99.55 ± 0.07	92.21 ± 0.21	95.60 ± 0.22
no	99.54 ± 0.06	92.16 ± 0.48	95.49 ± 0.15

Table C.4: Test accuracy of the pre-trained models.

C.3.3 Feature space attack

In Table C.5, we provide a more detailed numerical supplement to Figure 8.6. This shows results for a standard model, rather than with low-pass filters inserted between the residual blocks. Consistent with Figure 8.6, we observe that for standard models spectral normalisation tends to increase the distance in feature space distance of this attack, but we are unable to observe the low-frequency/high-frequency delta predicted by our theory without explicitly enforcing the domination assumption, suggesting that the adversarial search is able to find counter-examples to this condition.

	All freq		High freq		Low freq	
Lip(g)	$\|x - x_0\|$	$\|y - y_0\|$	$\|x - x_0\|$	$\|y - y_0\|$	$\|x - x_0\|$	$\|y - y_0\|$
0.9	4.47 ± 0.00	0.89 ± 0.06	4.47 ± 0.01	0.90 ± 0.06	4.45 ± 0.02	0.63 ± 0.05
0.95	4.47 ± 0.01	0.87 ± 0.03	4.47 ± 0.00	0.89 ± 0.03	4.47 ± 0.02	0.63 ± 0.03
0.99	4.47 ± 0.00	0.92 ± 0.06	4.47 ± 0.01	0.92 ± 0.06	4.48 ± 0.01	0.64 ± 0.03
3.0	4.47 ± 0.00	0.75 ± 0.04	4.47 ± 0.00	0.76 ± 0.04	4.47 ± 0.03	0.55 ± 0.04
6.0	4.47 ± 0.00	0.71 ± 0.05	4.47 ± 0.00	0.72 ± 0.06	4.46 ± 0.03	0.50 ± 0.04
9.0	4.47 ± 0.01	0.62 ± 0.02	4.47 ± 0.00	0.63 ± 0.03	4.47 ± 0.01	0.45 ± 0.02
no	4.48 ± 0.01	0.67 ± 0.04	4.47 ± 0.00	0.69 ± 0.04	4.46 ± 0.02	0.48 ± 0.02

Table C.5: Feature space movement of the adversarial attack on CIFAR10.

C.4 Architectural And Training Details

We make several small modifications to the standard ResNet architecture in order to make our theoretical analysis easier to verify. Firstly, we move BatchNorm from the residual branches and place it between residual blocks instead, as we found that regularising BatchNorm to have Lipschitz < 1, while fairly easy to implement (Gouk et al., 2021), was difficult to train because this significantly changes the dynamics of how BatchNorm is supposed to behave. It is common to use a strided 1x1 convolution on the skip connection when the residual branch also has a downsampling operation. This goes against our assumption that the residual connection is an identity. However, a residual connection which simply performs downsampling with BlurPool is an identity on the low frequency components of the input. Therefore, in order to make the structure of the network conform to our analysis, we replace the 1x1 convolution on the skip connection with BlurPool operation. To handle the increasing number of channels, we use zero padding. This operations is isometric (that is, $||x - y|| = ||f(x) - f(y)|| \forall x, y$), and so all proofs apply. These changes make it simpler to verify our theoretical conditions by comparing distances between feature maps before and after the residual connection. We use BlurPool in all convolution layers that perform downsampling, as described in (Zhang, 2019).

We use the standard train/test splits for all datasets used. We use SGD with momentum 0.9 and a learning rate schedule to train all models; the initial learning rate is 0.1. On CIFAR10, we divide this by 10 at epochs 150 and 250 and train for 350 epochs in total. For MNIST and FashionMNIST, we train for 200 epochs and divide the learning rate by 5 at epochs 60, 120 and 160. These are following the schedules used in (Mukhoti et al., 2021). We do this to produce trained models which can reasonably be said to be representative; though our accuracy and AUROC are not SotA, they are comparable to the results achieved in the literature.

We trained our models on an internal cluster, using a mix of GeForce 1080's, 2080's and Titan RTX's. We did not record the exact amount of compute time used, but it was relatively modest - on the order of a few GPU days to generate all listed results.