

Understanding Approximation for Bayesian Inference in Neural Networks

DPhil Thesis



Sebastian Farquhar
Balliol College
University of Oxford

`sebastian.farquhar@cs.ox.ac.uk`

Supervised by Yarin Gal

2022

Acknowledgements

I would like to thank Yarin for helping me become a scientist. One of our first conversations was about how the opportunities created by empirical phenomena in machine learning in this decade were every bit as exciting as phenomena in electricity and magnetism of the nineteenth century. That spirit of exploration has stuck with me, and become an essential part of how I see myself and my work, though we may never get any Maxwell's equations.

In addition, I am grateful to many others for the intellectual stimulation, mentorship, or friendship they have offered during my studies, especially (in alphabetical order): Milad Alizadeh, Joost van Amersfoort, David Burt, Adam Cobb, Tom Everitt, Angelos Filos, Andrew Foong, Aidan Gomez, Andreas Kirsch, Jannik Kossen, Yingzhen Li, Clare Lyle, Romy Minko, Mike Osborne, James Pavur, Tom Rainforth, Tim Rudner, Lewis Smith, and Mark van der Wilk. In some cases they did things for me that they may have considered small, but meant a great deal to me.

I would also like to thank Professor Stephen Roberts and Professor Andrew Gordon Wilson for serving as examiners of this thesis. Their expertise and diligence have greatly improved this thesis and more importantly have significantly changed how I think about these issues.

My intellectual progress has also depended essentially on the generosity of many anonymous reviewers. I wish I could express my gratitude to them personally. If you happen to have reviewed one of my papers and are reading this, I want you to understand that my appreciation is sincerely meant.

My brother, Gregory Farquhar, has also been a constant source of encouragement and an invaluable sounding-board for my research ideas. I think that my favorite paper so far only exists because he encouraged me to just spend a fortnight trying to get something solid enough to persuade an initially-skeptical-Yarin it was worth pursuing further.

Besides the research of the last few years, I want to thank both of my parents for teaching me the habit of trying to understand what I see and say. My father, Adam Farquhar, especially deserves credit for helping me to learn pragmatic idealism. My mother, Angela Dappert, taught me so much that I value, but among them was the perseverance she showed in her own journey to getting a PhD, despite innumerable obstacles. It saddens me unspeakably that she will never see my completed thesis, and I miss her every day.

I would especially like to thank my wife, Leah Broad, for her kindness, love, and guidance. Her support makes it possible for me to read what Reviewer 2 has to say for the truths it contains without fear or anger. Beyond that, her diligence and determination are an inspiration to me. It does not surprise me one jot that she beat me in our race as to whether she would hand in her book before I handed in my thesis.

Abstract

Bayesian inference has theoretical attractions as a principled framework for reasoning about beliefs. However, the motivations of Bayesian inference which claim it to be the only ‘rational’ kind of reasoning do not apply in practice. They create a binary split in which all approximate inference is equally ‘irrational’. Instead, we should ask ourselves how to define a spectrum of more- and less-rational reasoning that explains why we might prefer one Bayesian approximation to another. I explore approximate inference in Bayesian neural networks and consider the unintended interactions between the probabilistic model, approximating distribution, optimization algorithm, and dataset. The complexity of these interactions highlights the difficulty of any strategy for evaluating Bayesian approximations which focuses entirely on the method, outside the context of specific datasets and decision-problems. For given applications, the expected utility of the approximate posterior can measure inference quality. To assess a model’s ability to incorporate different parts of the Bayesian framework we can identify desirable characteristic behaviours of Bayesian reasoning and pick decision-problems that make heavy use of those behaviours. Here, we use continual learning (testing the ability to update sequentially) and active learning (testing the ability to represent credence). But existing continual and active learning set-ups pose challenges that have nothing to do with posterior quality which can distort their ability to evaluate Bayesian approximations. These unrelated challenges can be removed or reduced, allowing better evaluation of approximate inference methods.

Contents

1	Summary	13
1.1	Evaluating Bayesian Approximations	13
1.2	Approximating Distributions Suit Different Architectures	14
1.3	Optimizers Suit Different Approximating Distributions	15
1.4	How Good Is My Bayesian Approximation?	16
1.5	Overview	17
2	Challenges for Evaluating Approximate Bayesian Inference in Neural Networks	21
2.1	Ideal Bayesian Inference	22
	Axioms for Reasoning	22
	Dutch Books	23
	Bayes and Probability	23
	Bayes for Machine Learning	24
	Bayesian Model Comparison	25
2.2	Common Approximations to Bayesian Inference	26
	Approximating the Model	26
	Parametric Approximate Bayesian Inference	27
2.3	Conceptual Challenges for Evaluating Approximate Bayes	30
3	Network Parameterization Determines True Posterior and Approximation	33
3.1	True Posterior Correlations	36
	Methodology	36
	Results	38
3.2	Are There ‘Good’ Approximate Posteriors?	39
3.3	Do Approximate Inference Methods Find Good Mean-field Ap- proximations?	44
	Emergent Covariance in Deep Linear Mean-Field Networks . . .	44
	Product Matrices in Piecewise-Linear Mean-field BNNs	48
	Does Mean-field Approximate Inference Work Empirically? . . .	51
3.4	Discussion	54
4	Approximation Assumptions Affect Optimization	55
4.1	Understanding the Soap Bubble	56
	The Soap Bubble In Variational Inference	57
4.2	Overcoming the Soap Bubble	58

Evaluating the Objective	60
4.3 Diabetic Retinopathy Pre-screening	61
4.4 Discussion and analysis	67
MFVI and Radial variance	68
MFVI problem	68
MFVI mechanism	69
Avoiding the Pathology in MFVI	69
4.5 Conclusion	71
5 Evaluating Bayesian Deep Learning	73
5.1 Continual Learning	74
The Continual Learning Challenge	74
Continual Learning as Bayesian Sequential Learning	76
Inadequate Evaluations of Continual Learning	77
Good Evaluations of Bayesian Deep Learning with Continual Learning	81
Contrast to Inadequate Evaluations	82
5.2 Active Learning	84
Bias in Active Learning	85
Active Learning Is An Ineffective Posterior Evaluation	88
Active Learning Bias in the Context of Overall Bias	90
Related Work on Unbiased Active Learning	95
5.3 Conclusion	95
6 Conclusion and Discussion	99
Bibliography	105
A Appendix to Chapter 3	119
A.1 Experimental Details	119
Full Description of Covariance Visualization	119
Effect of Depth Measured on Iris Experimental Settings	119
HMC Experimental Settings	120
Diagonal- and Structured-SWAG at Varying Depths	123
A.2 Proofs	123
Full Derivation of the Product Matrix Covariance	123
Matrix Variate Gaussian as a Special Case of Three-Layer Prod- uct Matrix	127
Proof of Linearized Product Matrix Covariance	128
Existence Proof of a Two-Hidden-Layer Mean-field Approximate Posterior Inducing the True Posterior Predictive	133
B Appendix to Approximation Assumptions Affect Optimization	143

B.1	Derivation of the Entropy Term of the KL-divergence	143
B.2	Setting a Radial Prior	146
B.3	Experimental Settings	148
	Diabetic Retinopathy Settings	148
C	Appendix to Evaluating Bayesian Deep Learning	151
C.1	Experimental Details	151
	Linear Regression	151
	Bayesian Neural Network	153
C.2	Deep Active Learning In Practice	155

List of Figures

3.1	Parameter density histogram from HMC	37
3.2	‘Cost’ of the mean-field approximation relative to full-covariance for approximate Gaussian	39
3.3	1-layer BNN shows little ‘in-between’ uncertainty.	43
3.4	3-layer BNN shows significant ‘in-between’ uncertainty (but not as much as HMC).	43
3.5	Mean-field layers induce rich covariance product matrices	46
3.6	Product matrix elements can approximate Gaussians	48
3.7	Induced product matrices can have multimodal densities	49
3.8	Mean-field matches full-covariance after two layers on Iris	52
3.9	CIFAR-10. Diagonal and low-rank SWAG show similar performance	52
3.10	CIFAR-100. Diagonal and low-rank SWAG show similar performance	52
4.1	Comparison of the radial densities of high-dimensional multivariate Gaussians and our distribution	56
4.2	Visualizing the soap bubble	57
4.3	The Radial Gaussian distribution is much more sharply peaked and has lighter tails than a Gaussian.	60
4.4	Example retinopathy images	62
4.5	Hyperparameter robustness on diabetic retinopathy	65
4.6	Diagnosing high variance estimators of the optimization target using a truncated proxy	68
4.7	Diagnosing optimization failures for multivariate Gaussians	70
5.1	Permuted MNIST is an unrealistically dissimilar continuation	79
5.2	Single-headed Split set-ups reveal the failure of Bayesian sequential updating to prevent catastrophic forgetting.	82
5.4	Illustration of active learning in an unbiased way	87
5.5	Active learning is an ineffective evaluation of posterior quality	89

5.6	In some cases, the effect of debiasing is small enough to allow active learning to reveal performance differences.	91
5.7	Active evaluation can be used as a way to evaluate approximate inference performance.	92
5.8	Unbiased active learning evaluation	93
5.9	Overfitting bias can cancel with the bias introduced by active learning	96
A.1	Visualizing local product matrices	130
A.2	Structure of the UAT argument	134
C.1	Variance of the estimators	151
C.2	Epsilon-greedy proposal distributions	152
C.4	Using unbiased estimators throughout acquisition and training . .	158
C.5	Active learning plots using standard error to demonstrate differences	158
C.6	Temperature ablation for unbiased active learning	159
C.7	Further unbiased active learning downstream performance ablations	159

List of Tables

3.1	Time complexity of mean-field, full-covariance, and structured-covariance approximations	35
3.2	Imagenet. Mean-field and structured-covariance methods show similar performance.	52
3.3	CIFAR-10. Mean-field and structured-covariance methods show similar performance.	53
4.1	Diabetic retinopathy prescreening performance evaluation	65
4.2	Comparison of mean-field VI and Radial Bayesian neural networks on UCI datasets	66
A.1	Hyperparameters for toy regression visualization	120
A.2	Hyperparameters for covariance visualization	121
A.3	Hyperparameters for full-covariance experiment	122
A.4	HMC diagnostics for ReLU networks	122
C.1	Hyperparameters for active MNIST experiments	153
C.2	Overview of active learning bias in prior work	155

Chapter 1

Summary

1.1 Evaluating Bayesian Approximations

Bayesian inference has many attractions. When we know the structure of the distribution generating our data, and it allows exact inference, probabilistic updates to prior beliefs let us elegantly compress what we know and assess how ignorant we are. In that happy case, Bayesian inference arguably follows directly from axioms that represent basic rationality principles [Cox, 1961, Jaynes, 2003] and doing anything else can be exploited to your disadvantage [Ramsey, 1926, de Finetti, 1937]. There are reasonable criticisms even in this case, but the conceptual foundations of Bayesian inference are at their strongest.

Unfortunately, in practice we need to make large approximations. We need to restrict the class of models that we examine and to make approximations to exact inference in those models. Insofar as Bayesian inference is entailed by principles of rationality, following arguments like those of [Jaynes, 2003], the principles only offer a binary split of the world into ‘rational’ and ‘not rational’. Approximate Bayesian inference, the only kind anybody does, is solidly in the ‘not rational’ category. But this really just demonstrates how unhelpful that narrow notion of ‘rationality’ is. The sorts of axiomatic or Dutch-book motivations of Bayesian inference do not really give us a way of deciding which kinds of automated reasoning we should prefer when choosing between practical options.

Methods inspired by Bayesian inference which make use of the approximate probabilistic inference can be very successful. Moreover, they offer a range of capabilities that ‘non-Bayesian’ methods do not necessarily: such as the ability to ‘introspect’ and know what the model does not know.

So, what makes one Bayesian approximation better than another one? I argue that it is not a matter of adherence to Bayesian doctrine. The complicated interactions of the architecture of the model, the choice of approximate inference technique, the dataset, and the optimization strategies make it almost impossible to tell up-front which of two methods actually *is* more ‘principled’. An approximation that appears to be severe in one setting can be very mild in another, and vice-versa. This thesis explores two settings where this sort of behaviour arises, and the significance of different kinds of approximation can vary greatly depending on the context.

In contrast, I try to answer the question of how to evaluate Bayesian approximations in a pragmatic, and largely largely frequentist, way—good expected utility of the approximate posterior under cross-validation. In order to do this, we want to: identify desirable characteristic behaviours of Bayesian systems, identify decision-problem families that rely primarily on that characteristic behaviour, and develop evaluations for those decision-problems that test the ability of approximate Bayesian methods to provide the desired characteristic behaviour and which generalize to other examples of that decision-problem family. We can then say that the quality of the Bayesian approximation lies in its ability to deliver the desired characteristic behaviours in a wide range of settings. This will never be a guarantee—the behaviour might change in settings that appear to be very similar—but by operationalizing the things we want from approximate Bayesian methods we can get firmer purchase on the idea of what makes an approximation ‘good’ than we can manage otherwise.

This thesis proposes two examples of a characteristic behaviour we want from Bayesian approximations. First, the ability to sequentially update approximate posteriors in light of new information. Second, the ability of an approximate posterior to represent uncertainty at a parameter-level in a way that lets us know what information would reduce our uncertainty the most. This points towards decision-problem families of continual learning and active learning respectively (although other decision-problem families might be just as suitable for evaluating these characteristic behaviours).

1.2 Approximating Distributions Suit Different Architectures

The first example of the complicated interactions between architecture and approximation method which we explore (chapter 3) focuses on methods that pick an approximating distribution over a parametric model. Rather than allowing our inference to give us a fully general distribution, one often insists on some simplifying constraints: perhaps our parameters must all follow Gaussian distributions [MacKay, 1992c] or be independent of each other [Hinton and van Camp, 1993]. Many authors discuss the restrictiveness of these as-

sumptions in absolute terms (including MacKay [1992c]). But this misses the interaction between these approximating distributions and the ways that they are *used*.

Imagine, for example, how limited a uniform distribution is as a way to express a posterior distribution. But by being slightly clever about how we *use* these uniform distributions, we can easily express a Gaussian distribution [Box and Muller, 1958]. In an analogous way, though much more complicated, a neural network which is parameterized with very simple approximating distributions can express very complex distributions over *predictions*. Only three arbitrarily wide layers are required to allow a multi-layer perceptron to express arbitrary predictive distributions, under otherwise mild assumptions. In practice, methods including variational inference [Jordan et al., 1999] and mean-field-ammortized stochastic-gradient Markov Chain Monte Carlo [Maddox et al., 2019] seem to be remarkably unaffected by the mean-field assumption, and this may be because they find good posterior predictive distributions from a suitable mode of the parameter posterior.

As a result, existing efforts to enrich the approximating distributions over parameters may be misguided for certain tasks (in cases where the actual distributions over parameters *do* matter, such as transfer learning [?] or continual learning [Ritter et al., 2018], this is not true). In practice a simple approximating distribution in a complicated network is often easier to work with, and computationally cheaper, than a complicated distribution in a simple parametric model.

1.3 Optimizers Suit Different Approximating Distributions

The second example considered in this thesis (chapter 4) of unexpected difficulties when understanding how ‘good’ a Bayesian approximation comes from the interaction between the optimizer and approximating distribution in stochastic variational inference in neural networks. Where the architectures used in modern neural networks suit independent Gaussian distributions over parameters very well, the most common choices of optimizer may not. Stochastic gradient-based optimizers can be used effectively for variational inference in Bayesian neural networks with independent Gaussian approximating distributions [Blundell et al., 2015]. But researchers have found difficulties in using this approach in large neural networks without applying a number of tweaks which undermine the Bayesian principles underlying the methods.

Part of the reason for this lies in the strange sampling properties of independent Gaussian random variables in high-dimensional spaces. In high-dimensions, with very high probability samples come from a hypersphere ‘soap-bubble’. This dispersion increases the variance of gradient estimators which are based on samples from the approximating distribution. This can be

corrected by using a slightly modified approximating distribution, a Radial-Gaussian distribution, which does not demonstrate the ‘soap-bubble’ pathology.

1.4 How Good Is My Bayesian Approximation?

Until now, we have loosely helped ourselves to notions of one approximate Bayesian posterior being ‘better’ than another, but this idea is hard to pin down. In practice, researchers often use metrics like accuracy or log-likelihood on a held-out test set as a way to compare the performance of approximate Bayesian models. This choice has some principled motivations from modelling a Bayesian decision problem [Key et al., 1999]. But even if we wanted to use the marginal likelihood instead, as suggested by [MacKay, 1992a], this would be intractable. An extension of the approach of Key et al. [1999] is to select a number of interesting applications that test parts of Bayesian approximations that we value, and examine the expected utility of the approximate posterior.

This approach shares motivations with loss-calibrated (approximate) inference [Lacoste-Julien et al., 2011, Cobb et al., 2018]. Loss-calibrated inference aims to produce posterior distributions that are tailored to a specific utility function. We are proposing using the degree to which an approximate posterior has high utility on a task that exemplifies a characteristic behaviour as an *evaluation* of good approximate inference. That is, we are not just trying to find approximate posteriors that maximize expected utility, we are trying to identify specific applications where the goal of maximizing expected utility will require achieving the desired characteristic behaviours.

Two such applications are continual learning and active learning. Continual learning, in a Bayesian fashion [Nguyen et al., 2018], involves using a Bayesian model to compress data observed up to a time, and then to sequentially update this model as new data arrives. In true Bayesian inference, the prior can be updated sequentially with each data point in any order and reach the same posterior distribution. Performing well at sequential learning therefore tests an approximation’s ability to capture this aspect of Bayesian inference, which is at least partly a measure of the approximation’s success compressing the available data. Active learning, in contrast, involves picking which unlabeled data points should be labelled in order to learn as effectively as possible. In a Bayesian setting, this problem can be interpreted as estimating expected information gain [MacKay, 1992b]. Good active learning performance, therefore, can be interpreted as showing that an approximate Bayesian model is able to identify places where the uncertainty is most capable of being reduced by new information.

However, in both of these cases, the application raises considerable difficulty interpreting performance measures. In the case of continual learning, deficiencies in standard benchmarks have hidden the failures of Bayesian meth-

ods. Desiderata can be identified for continual learning evaluations that are both more suited to applications and better measures of Bayesian approximation. For active learning, performance measures are complicated by the fact that active learning in practice relies on implicit biases. The most useful approximate posteriors during active learning are the ones that encode these implicit biases well, rather than the ones that come closest to the true posterior distribution of the given model. By removing the bias, we can flip the ordering that active learning evaluations assign to different approximate posteriors. Instead, we may be able to use active testing [Kossen et al., 2021] as an evaluation, which is not dependent on implicit biases in the same way (though active testing may face its own challenges).

1.5 Overview

In chapter 2, I consider the challenges for approximate Bayesian inference. I review the motivations for Bayesian inference in exact settings as well as the literature on various approximations. Immediately, many of the ways in which ‘good’ exact Bayesian inference was historically motivated become inapplicable. Measuring quality in terms of the expected utility on representative decision-problems for desirable characteristic behaviours of an approximate posterior remains a sensible option.

In chapter 3, I consider how the architecture of a neural network affects the family of predictive distributions which a parametric approximating distribution can represent. Using deeper neural networks can allow even simple mean-field Gaussian approximating distributions over parameters to approximate arbitrarily complicated predictive posterior distributions to arbitrary precision as they become arbitrarily wide. I investigate this phenomenon from several analytical angles, and suggest that in practice approximation methods seem to uncover these successful modes.

In chapter 4, I consider how parametric approximating distributions can affect optimization procedures used in some approaches to approximate Bayesian inference. Because the approximating distribution affects the sampling properties of the optimization objective, we can prefer approximating distributions which lead to low-variance Monte Carlo estimators of the loss, even if these were a worse fit to the true posterior than the global optimum for another approximating distribution. In particular, I show how avoiding pathologies of mean-field Gaussian distributions in high-dimensional spaces can improve variational approximations by reducing the variance of Monte Carlo loss estimators.

In chapter 5 we return to the use of expected posterior utility for model evaluation. Active learning and continual learning each test different and important aspects of Bayesian approximations. However, when used naively, each can give a very misleading picture of the effectiveness of the Bayesian

approximations which are used. For each, I examine some of these failures and suggest ways to use these applications to get a better understanding of the quality of the approximate posterior.

Notation

General notation

A	proposition
x	scalar random variable
\mathbf{x}	vector random variable
\mathbf{X}	matrix random variable
x	scalar
\mathbf{x}	vector
\mathbf{X}	matrix

Machine learning notation

\mathcal{D}	data
\mathbf{x}_i	i 'th input data point (generally vector)
\mathbf{y}_i, y_i	i 'th supervision data point (vector, scalar)
$\hat{\mathbf{y}}_i, \hat{y}_i$	i 'th prediction of a model (vector, scalar)
$\mathcal{L}(\mathbf{y}_i, \hat{\mathbf{y}}_i)$	Loss for the i 'th prediction

Neural network notation

θ	neural network parameters
f_θ	neural network parameterized by θ

Other notation

\mathcal{N}	Gaussian distribution
$D_{\text{KL}}(\cdot \parallel \cdot)$	Kullback-Leibler divergence
$\mathbb{E}_p[\mathbf{x}]$	Expectation of \mathbf{x} under distribution p
$\mathcal{O}(\cdot)$	Big- O

Acronyms

NN	neural network
BNN	Bayesian neural network
VI	variational inference
MFVI	mean-field variational inference
FCVI	full-covariance variational inference
ELBO	evidence lower bound
GP	Gaussian process
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
i.i.d.	independently and identically distributed
p.d.f.	probability density function
c.d.f.	cumulative distribution function

Chapter 2

Challenges for Evaluating Approximate Bayesian Inference in Neural Networks

How can we know if we are doing Bayesian inference well? In some sense, the answer is trivial. All we need to do is to select a model and prior that express our subjective beliefs and update it following the laws of probability.

In practice, this approach struggles. We select models and priors that we know how to express and choose them partly for computational simplicity. For complicated models like neural networks, we do not know what the stated priors actually mean—let alone whether they match our beliefs. In addition, following the laws of probability is computationally intractable for most models.

This creates the challenge which this thesis addresses. Can we construct evaluations which give us insight into how well our machine learning systems are pursuing Bayesian goals? Falling short of idealised Bayesian inference is guaranteed—our task is to define a spectrum of degrees of Bayesian success. Without making this precise, it is hard to know whether research in approximate Bayesian methods is making progress.

The first two sections of this chapter should be interpreted as an opinionated review of prior work. The first considers the foundational motivations for an unrealisable Bayesian ideal about finding the ‘right’ posterior distribution. The second discusses obstacles to using this idealised approach in practice. In the third section, I explain why aiming for an approximate posterior that is ‘close’ to the true posterior is the wrong way to evaluate the quality of approximate inference.

2.1 Ideal Bayesian Inference

Bayesian inference is a formal system for reasoning about beliefs. This makes it a promising candidate for understanding machine learning. With Bayesian inference, we represent the strength of our belief in various propositions as probabilities, and then adjust beliefs in light of new evidence by following the laws of probability. Why would we want idealised Bayesian inference?

Axioms for Reasoning

Jaynes [2003] develops an argument by Cox [1961] that Bayesian inference is not just *a* way to reason about information, it is the *only* rational way to do so. They prove that every system of reasoning that obeys some assumptions follows the laws of probability, which in turn entail the ‘updating’ procedure commonly associated with Bayesian inference. I sketch the assumptions and character of their argument, deliberately leaving the details aside.

Assume that reasoning is a procedure that lets us adapt the strength of our belief about the plausibility of claims as we accumulate evidence. Assume further that we can represent degrees of belief using numbers, where more plausible claims get bigger numbers and infinitesimally more plausible things get infinitesimally bigger numbers. These assumptions are not completely trivial, but they come easily to machine learning practitioners who are in the habit of assigning numbers to messy things.

Add some modest rules about how these degrees of belief behave. Suppose I think I saw you in the supermarket yesterday. But then you tell me that you were in a different city giving a talk. This makes it less likely that I saw you in the supermarket, but has no bearing at all on whether the sun will rise tomorrow. Our modest rules entail two things. First, on learning you were in a different city, I ought to think less likely the conjunction that I saw you in the supermarket and that the sun will rise. Second, on learning you were in a different city, I should think it more likely that I did *not* see you in the supermarket (since we know *a priori* that you cannot be in two cities at the same time). This can be easily formalized and is plausible.

Last, we add some consistency conditions. We assume that if any conclusion can be reached in multiple ways, all procedures of reasoning reach the same conclusion. We assume that we always use all our evidence. We assume that equivalent states of knowledge get assigned the same number.¹

In practice, none of these conditions describe how people actually reason. Moreover, it seems likely that no physically implementable system could satisfy all these conditions. Any definition of ‘rationality’ which is so demanding as to be impossible to implement does not correspond to natural use of the

¹This final assumption seems problematic to me. I have not seen a grounding of equivalent states of knowledge which seems non-circular and I believe this mostly reprises the same problems Laplace had in using symmetry to ground classical probabilities.

term. But there is a sense in which they might represent an ideal for how one ought to reason. This raises the question of how we ought to evaluate *degrees* of rationality along a spectrum where a system that follows these axioms is at the unattainable optimum.

Dutch Books

There is another way to give teeth to ideal Bayesian inference. A family of results [Ramsey, 1926, de Finetti, 1937], called Dutch Book arguments, operationalize the mistake of not following these probability rules. These arguments demonstrate that any system of reasoning which assigns numbers to degrees of belief about a proposition, which does not follow the laws of probability regarding the relationship between those numbers, can be taken advantage of. They show that if you follow such a ‘non-Bayesian’ system of reasoning, it is possible to construct bets such that if you set the price, and I am allowed to choose whether to buy or sell the bet, you are guaranteed to lose money no matter the outcome [Jackson and Pargetter, 1976]. Moreover, the converse holds and following the laws of probability protects you from having a Dutch book formed against you [Lehman, 1955, Kemeny, 1955]

Once again, we can quibble with the strength of these arguments. It’s a narrow sense of ‘rationality’, maybe. Perhaps it is just as rational to opt out, wherever possible, of letting strangers pick which side of a bet they are on after you declare a price!

But equally, the argument should give us pause. A Dutch book feels like a violation of the principles of our reasoning system in a way that can be separated from the pragmatics of forming bets with predatory strangers [Skyrms, 1987].

More than that, if we, as machine learning researchers, want to develop mathematical systems of reasoning that are *not* robust to this sort of betting game, but we want to deploy our systems in practice, then we should make very sure that the decision-problems we work on are not isomorphic to this sort of betting game. That is, when I roll out a face-recognition algorithm I ought to be sure that none of its users are doing something that is similar enough at a conceptual level that the same results apply. This is especially true when machine learning systems might encounter interested adversaries who might construct situations that create a Dutch Book-style dynamic. To save ourselves the difficulty of constructing our decision problems to avoid this, and proving that we have done so (if this is possible, which I am unsure of), we might find our lives greatly simplified by following the rules of probability.

Bayes and Probability

Suppose that degrees of belief are the sorts of things that you can apply the laws of probability to, and that we do so. How does this relate to Bayesian

inference?

Bayes' theorem follows from the product rule of probability and commutativity of conjunction. Consider some claims about the world, A and B . For example, A might be true if and only if I saw you at the supermarket yesterday. We assign a credence to this claim, representing its degree of plausibility, $P(A)$. We further define $P(A | B)$ to be the plausibility that A is true given the evidence provided by B . The product rule states that $P(AB) = P(A | B)P(B)$, where AB is used to stand for the probability that both A and B are true at once. The commutativity of probabilities states that $P(AB) = P(BA)$ (after all, A and B are true precisely when B and A are). Then

$$\begin{aligned}P(B | A)P(A) &= P(BA) = P(AB) = P(A | B)P(B) \\P(B | A)P(A) &= P(A | B)P(B) \\P(B | A) &= \frac{P(A | B)P(B)}{P(A)}. \quad (\text{Bayes' Theorem})\end{aligned}$$

There are multiple sets of laws of probability, such as those by Kolmogorov [1950] and Cox [1961]. While they disagree about some subtleties, especially involving infinitesimals, they agree about these rules, and so they agree about Bayes' theorem. Although abstract at this point, it offers a powerful rule for changing your mind about B when you learn about A .

A few notational remarks. Propositions are named with capital letters A , B , etc. In work on probability it is common to name random variables with capital letters. However, because it can be important in machine learning to distinguish scalar, vector, and matrix quantities, we adopt the practice of denoting scalar, vector, and matrix random variables using the letters like x , \mathbf{x} , and \mathbf{X} . Their realizations are denoted x , \mathbf{x} , and \mathbf{X} and are drawn from spaces which are defined where required but are typically calligraphic capitals (e.g., \mathcal{X}). Some special variables depart from these conventions for sake of consistency with other work, which will be noted as it arises.

Bayes for Machine Learning

Earlier, we chose to view machine learning as a formal system for reasoning about beliefs. Moreover, we wanted a way to move from observations to generalizations about what one might observe.

One way to describe those generalizations is to write down a parametric model. We will be mostly concerned with neural networks, which are a particularly versatile family of non-linear parametric models.

$$\begin{aligned}\mathbf{x} \in \mathcal{X} &: \text{input} \\ \mathbf{y} \in \mathcal{Y} &: \text{output} \\ f_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{Y} &: \text{parametric model (sometimes just written } f) \\ \boldsymbol{\theta} &: \text{parameters of the parametric model}\end{aligned}$$

In order to generalize beyond our observations $\mathcal{D} := \{(\mathbf{x}_i, \mathbf{y}_i) \mid 0 \leq i < N\}$ we would like to infer a degree of plausibility for each possible value of the parameters in the context of that parametric model's structure. In the notation of probability this may be written $p(\boldsymbol{\theta} \mid \mathcal{D}; f)$ and by Bayes' Theorem

$$p(\boldsymbol{\theta} \mid \mathcal{D}; f) = \frac{\overbrace{p(\mathcal{D} \mid \boldsymbol{\theta}; f)}^{\text{likelihood}} \overbrace{p(\boldsymbol{\theta}; f)}^{\text{prior}}}{\underbrace{p(\mathcal{D})}_{\text{marginal likelihood}}}, \quad (2.1)$$

where we have used the fact that the probability of the observations is unaffected by the parametric model's structure. In what follows, I will mostly suppress the context of the function, as this is often taken for granted. It is, however, very important when we consider model-selection and approximation, which enters into section 2.2.

Within the context of that parametric model, if we could state the three terms on the right-hand-side we could state the plausibility of any parameter value. The plausibility of any output, \mathbf{y} , given some input, \mathbf{x} , can then be found by integrating over all possible parameter values

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}) = \int p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) d\boldsymbol{\theta}. \quad (2.2)$$

For many useful parametric models, the likelihood is provided directly by the model. In principle, the prior can be written down *a priori*—relying only on our subjective beliefs when we frame the problem. The marginal likelihood is challenging because it requires integrating over $\boldsymbol{\theta}$.

In this way, Bayes' rule offers a principled way to capture what we know about the parameters of a model given some observations. Moreover, when the true observation-generating process can be expressed by $f_{\boldsymbol{\theta}}$, in the limit of infinite observations our beliefs about $\boldsymbol{\theta}$ will converge to reality Doob [1949]. Suppose the data are generated by a process which is exactly a parametric likelihood function $f_{\boldsymbol{\theta}^*}$. Then, in the limit of infinite data, N , when we approximate $\boldsymbol{\theta}^*$ by $\boldsymbol{\theta}$, the Bernstein-von Mises theorem can be used to show that the posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ tends towards the normal distribution $\mathcal{N}(\boldsymbol{\theta}^*, \mathcal{O}(\frac{1}{N}))$. This offers reassurance that Bayesian inference ought to approach the truth as we gather more data.

Bayesian Model Comparison

Another strength of Bayesian inference is that it guides which function one ought to select. Suppose, for example, that we are choosing between f and g . We might denote the probability that f is the true data-generating process by $p(f)$.

Conditioning on our observations then gives us the degree of plausibility of the model in light of our observations, $p(f | \mathcal{D})$. By Bayes' Theorem this can be written

$$p(f | \mathcal{D}) = \frac{p(\mathcal{D} | f)p(f)}{p(\mathcal{D})}. \quad (2.3)$$

Estimating $p(f | \mathcal{D})$ is difficult because both the model prior and the marginal likelihood are not obvious. However, when choosing between two models about which we are equally ignorant, we can simplify the calculation [MacKay, 1992a] by estimating the ratio of the probabilities of the two models

$$\begin{aligned} \frac{p(f | \mathcal{D})}{p(g | \mathcal{D})} &= \frac{p(\mathcal{D} | f)p(f)}{p(\mathcal{D} | g)p(g)} \frac{p(\mathcal{D})}{p(\mathcal{D})} \\ &= \frac{p(\mathcal{D} | f)}{p(\mathcal{D} | g)}, \end{aligned} \quad (2.4)$$

where we have assumed that the two models under consideration are equiprobable *a priori*.

This is a useful trick in principle, because it reduces model selection to the problem of computing the marginal likelihood of the data given the model. However, in practice even this can be difficult to compute for models of interest. The marginal likelihood ratio is also not valid for model comparison in cases where we do not, in fact, want to assume that the two models are equally plausible *a priori* and does not necessarily predict generalization error [MacKay, 1992a, Lotfi et al., 2022].

2.2 Common Approximations to Bayesian Inference

So far, we have assumed a best-case scenario setting aside pragmatic concerns. Unfortunately almost all practical settings require substantial approximations. These approximations can be divided into approximations in defining the problem setting (choice of likelihood function and prior) and computational approximations (computing the posterior). However, these choices are linked. Chapters 3 and 4 focus on specific challenges created by the interactions of these two categories of approximations. Chapter 5 considers strategies for evaluation that are helpful when approximations have been made of either kind.

Approximating the Model

We never know the structure of the true data-generating process with certainty. This means that, especially when we restrict our models so that we are able to write them down and compute them, we cannot guarantee that

our model is capable of expressing the true process. Instead, we would like to pick an approximation which is not too restrictive.

Fortunately, an approximate extension of the Bernstein-von Mises theorem applies even when the model cannot express the true data-generating process [Kleijn and van der Vaart, 2012]. When the parameters which we are optimizing belong to the wrong parametric model, g_ϕ , nevertheless the posterior $p(\phi \mid \mathcal{D})$ converges to the Gaussian $\mathcal{N}(\phi^*, \mathcal{O}(\frac{1}{N}))$ such that

$$g_{\phi^*} = \arg \min_{\phi} D_{\text{KL}}(f_{\theta^*} \parallel g_{\phi}). \quad (2.5)$$

When our parametric model is over-parameterized and expressive, this suggests that the error introduced due to the model failing to express the true process is unlikely to be large. Partly for this reason, this thesis focuses mostly on computational approximations and the interactions between computational approximations and modeling approximations.

There are also some efforts to address model-misspecification more directly. For example, Grunwald [2011] makes use of a ‘safe’ maximum *a posteriori* (MAP) estimator which is designed to handle misspecified settings well. While there may be some promise in these sorts of approaches, they depart sufficiently from the basic Bayesian procedure to require separate treatment.

Meanwhile, Key et al. [1999] propose what is effectively computing the expected utility of a Bayesian posterior as a solution to model-misspecification, which they motivate as a solution to a properly framed Bayesian decision problem. They additionally argue for the value of ‘generic’ utility functions like the log-likelihood as a proxy for application-specific utilities. Although this argument has received little vindication from theoreticians, it has received the ultimate compliment of being more-or-less the only way that modern Bayesian deep learning papers evaluate their approximate posteriors. While they argue for this in the context of model-misspecification, I will return to this solution in more detail when we consider evaluating *approximate* Bayesian inference.

Parametric Approximate Bayesian Inference

Neural networks [Rosenblatt, 1958] are a family of parametric models which apply both linear and non-linear ‘layers’ to an input. Their architectures are diverse, and will be discussed in detail within later chapters where important. In deterministic neural networks, the values of the parameters can be ‘learned’, for example through backpropagation [Linnainmaa, 1970].

From a Bayesian perspective, we can acknowledge subjective uncertainty over the parameters of a neural network. Given a prior, it is then possible to infer the parameter distribution over neural network weights. Bayesian neural networks (BNNs) are networks whose weights follow a posterior distribution inferred in this way [Tishby et al., 1989, Buntine and Weigend, 1991], or whose weights are approximately distributed as a posterior distribution inferred in this way.

There is no discussion in the literature of what sort of approximation suffices for a parametric model to ‘count’ as approximately Bayesian. This very question might reasonably be regarded as so vague as to be meaningless. We will return to it later but only answer it partially—by identifying some characteristically Bayesian reasoning properties and proposing tests for their presence.

If a neural network, f , has parameters, θ , then we can infer a posterior distribution over parameters given some observed data, $p(\theta \mid \mathcal{D})$ following eq. (2.1). In order to do this, we must be able to fix a prior, $p(\theta)$, and define the likelihood function $p(\mathcal{D} \mid \theta)$, both of which are generally feasible.

However, the marginal likelihood, $p(\mathcal{D})$, is intractable. It is constant w.r.t. θ , so we can find $p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)p(\theta)$ as an unnormalized distribution. But when we make predictions following eq. (2.2), integrating over this unnormalized distribution is also intractable.

Variational Inference

One alternative is variational inference [Jordan et al., 1999]. Instead of computing the posterior distribution over parameters, we instead define an approximating distribution, $q(\theta)$, and minimize a measurement of difference between this and the true posterior. ‘Variational inference’ is what we call this procedure when the distance is the Kullback-Leibler (KL) divergence between the approximating distribution and the true posterior, $D_{\text{KL}}(q(\theta) \parallel p(\theta \mid \mathcal{D}))$.

In fact, this KL-divergence is not directly tractable either because we do not have access to the posterior distribution. Instead, we optimize a bound. Barber and Bishop [1998] show in the context of neural networks that:

$$D_{\text{KL}}(q(\theta) \parallel p(\theta \mid \mathcal{D})) = \log p(\mathcal{D}) - \text{ELBO}(q, p), \quad (2.6)$$

where the evidence lower bound (ELBO) is defined

$$\text{ELBO}(q, p) = \mathbb{E}_{q(\theta)} [\log p(\mathcal{D} \mid \theta)] - D_{\text{KL}}(q(\theta) \parallel p(\theta)). \quad (2.7)$$

Because the marginal likelihood is constant w.r.t. θ , minimizing the negative ELBO is equivalent to minimizing the KL-divergence between the approximating distribution and the true posterior. This recovers the objective from earlier work based on minimum description lengths [Hinton and van Camp, 1993].

In neural networks, we can maximize the ELBO using gradient-based optimization of a Monte Carlo approximation of the integrals [Graves, 2011, Blundell et al., 2015]. The integration approximation can be refined, in certain settings, with further approximations [Kingma et al., 2015, Wen et al., 2018b, Khan et al., 2018, Wu et al., 2019]. At the same time, while Blundell et al. [2015] use a fully factored Gaussian approximating distribution, other approximating distributions can be chosen. For example, partially or hierarchically factorised distributions are preferable for some applications (e.g.,

[Kessler et al., 2021]). Gal and Ghahramani [2015] propose an even simpler approximating distribution with Bernoulli noise multiplied into hidden units. On the other hand, one could use more complicated distributions which also capture correlations between weights [Louizos and Welling, 2016] or apply normalizing flows [Rezende and Mohamed, 2015].

Laplace Approximation

Laplace’s method offers another approximation for the posterior of a neural network [Denker and leCun, 1991, MacKay, 1992c]. As before, we choose an approximating distribution, often assumed Gaussian with independent parameters. We then fit a Gaussian distribution centred on the maximum-a-posteriori parameter values.

First, we maximize directly the unnormalized posterior probability density over the parameters, $p(\boldsymbol{\theta} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$. In the case of an independent Gaussian prior, this is just optimization w.r.t. the negative log-likelihood loss with an L_2 -norm regularization [Goodfellow et al., 2016]. Second, we find the Hessian of the loss with respect to parameter-spaces

$$H_{ij} = \frac{\partial^2 \mathcal{L}}{\partial \theta_i \partial \theta_j}. \quad (2.8)$$

Assuming that the distribution around the maximum is approximately an independent Gaussian, the variance in each dimension is given by the diagonal term of the Hessian. The independence assumption may be fully relaxed, or partially relaxed—for example, using matrix-variate Gaussian structured covariance approximations [Ritter et al., 2018].

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods can be used to draw samples from the approximate posterior distribution of neural networks [Neal, 1993]. Mackay [2003, Chapter 32] provides an excellent overview of MCMC methods as a whole. MCMC methods have the advantage that, in the limit of infinite samples, they draw samples from the true posterior distribution. In particular, this is likely to be an advantage when true posterior distributions are very unlike a Gaussian (say). For example, a multi-modal posterior might in principle be much better approximated by MCMC. However, they suffer from several disadvantages.

- Although they give provably good samples given sufficient steps, there is no way to know for sure that sufficient steps have been taken.
- For multi-modal distributions, it can take a very long time to ‘hop’ from one mode to another.

- Especially in high-dimensional spaces, exploration can be slow.
- The outputs are individual posterior samples. In deployment, these must be separately stored and loaded in and out of memory to compute eq. (2.2). This is very slow compared to drawing multiple samples from the same multivariate Gaussian distribution.
- Because the outputs are individual posterior samples, they are hard to use as a prior in sequential or online learning.
- Samples are locally correlated which can create problems in some applications (e.g., bounding estimator variance).

For neural networks specifically, a number of adaptations improve exploration as well as performance in the high-dimensional parameter spaces. Stochastic gradient Markov chain Monte Carlo (SG-MCMC) takes advantage of noise in stochastic gradient-based optimizers to perform the Markov chain stepping efficiently [Welling and Teh, 2011]. Hamiltonian dynamics can be included [Chen et al., 2014] and adaptive step-size schemes can improve efficient exploration [Hoffman and Gelman, 2014]. The drawback of memory and storage can also be addressed by amortizing the samples into a Gaussian distribution [Maddox et al., 2019].

As a result, in recent years some work has attempted to approximate large neural network posterior distributions using Hamiltonian Monte Carlo [Izmailov et al., 2021]. Although experimental HMC samples cannot be equated with the true posterior distribution they represent the best strategy currently available for high-fidelity inspection of the true posterior distribution.

Other Methods

A number of other methods to approximate Bayesian inference in neural networks exist. These include expectation propagation [Minka, 2001, Jylänki et al., 2014], probabilistic back-propagation [Hernández-Lobato and Adams, 2015], alpha-divergence minimization [Hernández-Lobato et al., 2016], and sequential Monte Carlo [Doucet et al., 2001]. However, because these are not currently used to a significant degree, I do not cover them in detail here. They are covered in more detail in other reviews of the literature for approximate Bayesian inference in neural networks [Gal, 2016, Alquier, 2020].

2.3 Conceptual Challenges for Evaluating Approximate Bayes

Section 2.1 discussed motivations for Bayesian inference through rationality considerations and Dutch books. Unfortunately, these binary motivations do not give a way to express being more or less rational.

This is a problem for the approximate Bayesian. Even if we assume that the true data-generating distribution is within the support of our probabilistic model, our approximations mean that we do not expect to actually produce the true posterior distribution. This means that we have not, in fact, followed the ‘rational’ rules of probability. Nor does it defend us against Dutch books. This has been termed the ‘approximate inference conundrum’ [Ghahramani, 2008].

It is natural to try to provide a partial answer by considering the distance between an approximate posterior and the true Bayesian posterior, in much the same way as variational inference is motivated. However, unfortunately, there are a number of plausible candidates. To name a few:

- the forwards KL-divergence (expectation propagation);
- the backwards KL-divergence (variational inference);
- the Wasserstein distance between the approximate and true posterior;
- the maximum density difference between the approximate and true posterior.

And for each of these we could consider the distance in parameter-space, function-space, or predictive-space. It is not clear to me, *a priori*, that any of these is the ‘correct’ sense of distance to the true posterior distribution. Moreover, it is clearly true that the optimal parameter distribution on one of these distances will not be optimal on another. For example, if the true posterior is multi-modal, the optimal approximations on the forwards and backwards KL-measures are radically dissimilar (one fits a single mode and the other covers them all). This means that our inability to select a ‘correct’ sense of distance also robs us of an ordering over approximate posterior quality.

It is my (unproven) belief that we will not be able to solve this problem without adding a few extra tools: a utility function and a data-distribution. These allow us to frame our problems as Bayesian decision problems. On this picture [Key et al., 1999], we care about the expected utility of the approximate posterior distribution for the problem of interest

$$\mathbb{E}[U] = -\mathbb{E}_{\mathbf{y}, \mathbf{x} \sim p_{\text{data}}, \boldsymbol{\theta} \sim q(\boldsymbol{\theta}; \mathcal{D})} \left[\int p(\hat{\mathbf{y}} | \mathbf{x}, \boldsymbol{\theta}) \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) d\hat{\mathbf{y}} \right]. \quad (2.9)$$

Here, we have specialized the utility, U , to be the negative model loss, $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$, for an actual \mathbf{y} and a predicted $\hat{\mathbf{y}}$. This is appropriate for decision-rules based on predictive functions, which we will focus on. The expectation is over both the data-distribution, p_{data} , and the parameter distribution, $q(\boldsymbol{\theta}; \mathcal{D})$ and represents the expected utility of our model for the problem we care about. Key et al. [1999] proposed this expected utility as a solution to the model-misspecification problem. In chapter 5 we will consider how this approach can

be extended to a more general family of approximations including computational approximations.

A key assumption is that the actual decision problem we will face is in the context of data generated from the same distribution as that which \mathcal{D} was drawn from. This ‘distribution matching’ assumption is more often associated with frequentism than Bayesian methods. However, for a Bayesian who cares about anything remotely practical this distinction is a mistake. In practice, it is never *important* to infer a posterior distribution for parameters except in the hope of generalizing to some other data. Even when we just want to explore data out of curiosity, it is our expectation that the data is related in some generalizable way to something else in the universe that makes the exploration meaningful. So a Bayesian cannot escape some sort of assumption connecting the data which have been observed to some problem of significance.

A further assumption is that we can write down a loss or utility function that is what we actually care about. A common choice, echoing Key et al. [1999] is the cross-entropy or negative log-likelihood loss which can supposedly stand in as a *universal* utility function. Note that almost all current work in approximate Bayesian inference in practice uses the cross-validation expected utility of the approximate posterior as a main assessment.

However, we also care about *using* Bayesian approximations for various downstream tasks. In my view, the philosophical commitment of a Bayesian approach is that probability distributions represent strengths of beliefs in states of affairs. This means that a Bayesian approach should offer introspection and compression. ‘Introspection’ in the sense that our posterior distribution ought to offer guidance about which parts of our knowledge are most uncertain and, at least in principle, most amenable to correction in light of new observation. ‘Compression’ in the sense that Bayesian inference ought to support sequential inference by representing the total state of belief after incorporating some evidence. The posterior from one set of observations can become a prior when approaching another dataset. Both of these aspects appear within the axioms of constructive formulations of both objective and subjective Bayesianism.

As a result, in chapter 5 we will look at two applications that most ideally characterize these two aspects of Bayesian inference: active learning and continual learning. It is difficult to structure these tasks in a way that genuinely tests the quality of the approximate posterior in some generalizable sense, rather than too narrowly evaluating unimportant details of the application. However, some progress can be made. In this way, a utility-centred framework for evaluating approximate posterior distributions can step past a reliance on purely general utility functions and start to evaluate different aspects of the Bayesian promise separately.

Chapter 3

Network Parameterization Determines True Posterior and Approximation

Notational remark: Because the proof of proposition 1 depends heavily on distinctions between random variables and realizations, but very little on distinctions between scalars/vectors/matrices, we adopt the standard notation from statistics that lower case letters are the realizations of upper case random variables, departing from the notation of other sections of the paper, such as the proofs of proposition 3 where the matrix structure is key.

While performing approximate Bayesian inference in Bayesian neural networks (BNNs) researchers often make what they regard as a severe approximation. This is variously called the ‘mean-field’ or ‘diagonal’ approximation. It assumes that the approximate posterior distribution over each weight is independent of all the others and such that the distribution fully factorizes in the following way:

$$q(\boldsymbol{\theta}; \mathcal{D}) = \prod q(\theta_i; \mathcal{D}). \quad (3.1)$$

This is called a diagonal approximation because the covariance matrix for the vector random variable $\boldsymbol{\theta}$ is diagonal. It is called mean-field by analogy to the mean-field solution to Ising models, in which local correlations between neighboring spins are ignored. In particular, a further approximation is often made that $q(\theta_i; \mathcal{D})$ is the Gaussian $\mathcal{N}(\mu_i, \sigma_i^2)$. This approximation has been widely applied in neural networks for example for variational inference [Hinton and van Camp, 1993, Graves, 2011, Blundell et al., 2015] or amortized SG-

MCMC [Maddox et al., 2019], although it is sometimes useful to make different factorisation assumptions based on hierarchical structures, for example Kessler et al. [2021].

Although they have often used the mean-field approximation, researchers have assumed that it is a severe limitation. After all, while we might choose to restrict the *approximate posterior* in this way, we rarely actually believe it to be true of the *true posterior*. In an influential paper on approximate Bayesian inference for neural networks [MacKay, 1992c], David MacKay framed the problem for the field, writing

The diagonal approximation is no good because of the strong posterior correlations in the parameters.

This has motivated extensive exploration of approximate posteriors that explicitly model correlations between weights. For example, for variational inference Barber and Bishop [1998] use a full-covariance Gaussian while later work has developed more efficient structured approximations [Louizos and Welling, 2016, Zhang et al., 2018, Mishkin et al., 2019, Oh et al., 2019]. Still further work has tried to significantly increase the expressive power of the approximate posterior distribution [Jaakkola and Jordan, 1998, Mnih and Gregor, 2014, Rezende and Mohamed, 2015, Louizos and Welling, 2017, Sun et al., 2019]. Similarly, for the Laplace approximation full- [MacKay, 1992c] and structure-covariance [Ritter et al., 2018] have been employed and structured-covariance for amortized SG-MCMC [Maddox et al., 2019].

In addition to these research efforts, Foong et al. [2020] have identified pathologies in mean-field approximate posteriors for neural networks with a single hidden-layer used for regression, and have conjectured that these might exist in deeper models as well and might apply in other contexts.

This chapter interrogates the prevailing assumption that mean-field methods are overly restrictive. The substance of MacKay’s objection to using diagonal approximate posteriors seems to have been an argument something like this: empirically, there are strong correlations between parameters in the true posterior distribution. If we want a good approximation, our approximate posterior distribution needs to be able to capture all the important properties of the true posterior, one of which is the presence of off-diagonal correlations. So a diagonal distribution will make a low-quality approximate posterior. Our investigation will mirror the steps of this argument:

- Does the *true* posterior distribution have strong correlations between the parameters?
- Do there exist approximate posterior distributions that are ‘good’ even if they are mean-field?
- Do actual methods for approximate Bayesian inference uncover these ‘good’ approximations?

	Complexity	
	Time	Parameter
Mean-field VI [Hinton and van Camp, 1993]	K^2	K^2
Full-covariance VI [Barber and Bishop, 1998]	K^{12}	K^4
Matrix-variate Gauss. [Louizos and Welling, 2016]	K^3	K^2
MVG-Inducing Point [ibid.]	$K^2 + P^3$	K^2
Noisy KFAC [Zhang et al., 2018]	K^3	K^2
SWAG [Maddox et al., 2019]	K^2N	K^2N

Table 3.1: Complexity of a forward pass in K —the number of hidden units for a square weight layer. Mean-field VI has better time complexity and avoids a numerically unstable matrix inversion. Inducing point approximations can help, but inducing dimension P then becomes a bottleneck. The structured-covariance approximation in Maddox et al. [2019] is in terms of N , the number of iterates used for computation, which can be *much* smaller than K (e.g., 20) but is not suitable for VI-based methods.

In doing so, we will explore the ways in which neural network *architectures* interact with the choice of approximating distribution and the ways in which an approximate posterior can be good or bad.

For the first question, we will provide empirical evidence that some modes of the true posterior for BNNs are approximately mean-field, especially in deeper networks. This suggests that a mean-field approximate posterior might be a good approximation even directly in *parameter-space*. For the second, we will provide a proof that for networks with at least two hidden-layers it is possible to approximate an arbitrary *predictive* posterior distribution arbitrarily closely. For many applications, we care far more about the predictive distribution than we do about the distributions over parameters, which makes this perspective more useful for considering the restrictiveness of approximation assumptions. For the third, we will provide a theoretical construction demonstrating how neural networks might in practice develop patterns that allow ‘good’ mean-field approximate posteriors and provide empirical evidence suggesting that this might emerge in practice. However, we do not prove or demonstrate that methods like variational inference in fact find these ‘nearly optimal’ predictive distributions. Despite this, there are things that a mean-field approximate posterior cannot capture, which makes most of our results contingent not only on the architecture under consideration but also the *dataset* and *problem*.

This matters for three reasons. First, the rejection of mean-field methods comes at a price. Relatively faithful structured covariance methods have worse time complexity (see table 3.1). Even efficient implementations take over twice as long to train an epoch as comparable mean-field approaches [Osawa et al., 2019]. In contrast parsimonious structured approximations such as the one used by Maddox et al. [2019] can have a minimal impact on time

and memory complexity (although it is not immediately clear how variational inference methods could incorporate this specific approximation). Researchers may benefit from exploring the most computationally feasible end of the spectrum of covariance-expressiveness rather than seeking ever-richer approximations. Second, getting to grips with the question of how suitable the mean-field approximation provides a valuable case study for understanding how to evaluate the quality of approximate posteriors generally. Third, the theoretical case against mean-field approximations has to handle some puzzles. Ordinary non-Bayesian neural networks are arguably an extreme case of the mean-field approximation and they often work very well in many applications. Moreover, recent work has succeeded in building mean-field BNNs which perform quite well (e.g., Khan et al. [2018], ?, Osawa et al. [2019], Wu et al. [2019], Farquhar et al. [2020]) or which perform well despite making even *more* stringent assumptions (e.g., Swiatkowski et al. [2020]).

The work in this chapter suggests that if a large amount of computational effort is required to approximate complicated posterior distributions then for many applications those resources might be better spent on using larger models with simpler Bayesian approximations instead.

3.1 True Posterior Correlations

Let’s consider the most basic question: does the true posterior distribution have strong correlations between the parameters? MacKay was working with very small neural networks with few parameters, and in this setting he did find evidence of correlations between weights.

These correlations might be less present in larger, deeper networks. Indeed, in later sections we will give some reasons to suspect that after three weight-layers—or two layers of hidden units—posterior correlations will be less necessary. We can explore posterior correlations in larger neural networks using Hamiltonian Monte Carlo (HMC), which is generally regarded as providing a higher quality of posterior sample than methods based on a parametric approximating distribution. This ‘microscope’ suggests that there are modes of the true posterior that are approximately mean-field. We examine a truly full-covariance posterior, not even assuming that layers are independent of each other, unlike Barber and Bishop [1998] and most structured covariance approximations.

Methodology

We use No-U-turn HMC sampling [Hoffman and Gelman, 2014] to approximate the true posterior distribution, $p(\boldsymbol{\theta} \mid \mathcal{D})$.¹ We then aim to determine

¹We use No-U-turn HMC in order to adaptively set the step-size and trajectory length in a way that is ‘fair’ to the different architectures which are considered here (as opposed

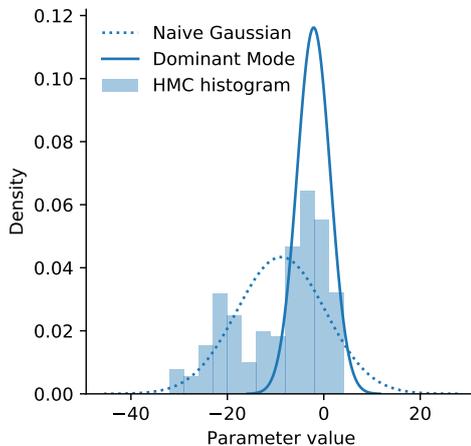


Figure 3.1: Example density for randomly chosen parameter from a ReLU network with three hidden layers. The HMC histogram is multimodal. If we picked the naive Gaussian fit, we would lie between the modes. By using a mixture model, we select the dominant mode, for which the Gaussian is a better fit.

how much worse a mean-field approximation to these samples is than a full-covariance one. To do this, we fit a full-covariance Gaussian distribution to the samples from the true posterior— $\hat{q}_{\text{full}}(\boldsymbol{\theta})$ —and a Gaussian constrained to be fully-factorized— $\hat{q}_{\text{diag}}(\boldsymbol{\theta})$.

We deliberately do not aim to sample from multiple modes of the true posterior—we are assessing the quality of approximation to a single mode by a Gaussian distribution—and for this reason it is less problematic than normal that HMC can struggle to explore multiple separated modes in high-dimensional spaces. A Gaussian is clearly unsuitable for approximating multiple modes at the same time, but a mixture of Gaussians can be used to approximate a multimodal posterior [Wilson and Izmailov, 2020]. Here, we fit a mixture of Gaussians to the HMC samples, using the Bayesian Information Criterion to select the number of mixture components, and then focus on the most highly weighted component for our analysis.² See appendix A.1 for details.

We consider two measures of distance:

1. **Wasserstein Distance:** We estimate the L_2 -Wasserstein distance between samples from the true posterior and each Gaussian approximation.

to picking settings on one depth or having to individually tune each). There is a growing literature on more efficient sampling for neural networks, including methods based on semi-separable HMC [Zhang and Sutton, 2014, Cobb et al., 2019a] and symmetric splitting [Cobb and Jalaian, 2021] which I did not use because public implementations were not available at the time of execution, but might have allowed better coverage of the posterior.

²The Bayesian Information Criterion was chosen for ease of implementation, but a more principled approach might have been to sample with a Dirichlet prior allowing shrinkage.

Define the Wasserstein Error:

$$E_W = W(p(\boldsymbol{\theta} \mid \mathcal{D}), \hat{q}_{\text{diag}}(\boldsymbol{\theta})) - W(p(\boldsymbol{\theta} \mid \mathcal{D}), \hat{q}_{\text{full}}(\boldsymbol{\theta})). \quad (3.2)$$

If the true posterior is fully factorized, then $E_W = 0$. The more harmful a fully-factorized assumption is to the approximate posterior, the larger E_W will be.

2. **KL-divergence:** We estimate the KL-divergence between the two Gaussian approximations. Define the KL Error:

$$E_{KL} = D_{\text{KL}}(\hat{q}_{\text{full}}(\boldsymbol{\theta}) \parallel \hat{q}_{\text{diag}}(\boldsymbol{\theta})). \quad (3.3)$$

This represents a worst-case information loss from using the diagonal Gaussian approximation rather than a full-covariance Gaussian, measured in nats (strictly, the infimum information loss under any possible discretization [Gray, 2011]). $E_{KL} = 0$ when the mode is naturally diagonal, and is larger the worse the approximation is.³

Either or both of these distances could increase even if either the ELBO or problem-specific loss function improved (c.f., discussion of the challenge for evaluating Bayesian approximations using the ‘distance’ to the true posterior distribution in section 2.3).

Each point on the graph represents an average over 20 restarts (over 2.5 million model evaluations per point on the plot). For all depths, we adjust the width so that the model has roughly 1,000 parameters and train on the binary classification ‘two-moons’ task. We report the sample test accuracies and acceptance rates in appendix A.1 and provide a full description of the method.

Note that we are only trying to establish how costly the *mean-field* approximation is relative to full covariance, not how costly the *Gaussian* approximation is. Our later results, however, will show that any target predictive distribution can be approximated arbitrarily closely by a sufficiently large neural network with a Gaussian approximate posterior.

Results

In fig. 3.2a we find that the Wasserstein Error introduced by the mean-field approximation is large in shallow fully-connected neural networks but falls rapidly as the models become deeper.⁴ In fig. 3.2b we similarly show that

³ $p(\boldsymbol{\theta} \mid \mathcal{D})$ is not directly involved in this error measure because KL-divergence does not follow the triangle inequality.

⁴We do not examine HMC samples from more sophisticated network architectures with convolutions, skip connections, or attention. This is a limitation of the analysis in this section as well as the linear and piece-wise linear analysis in the next section. However, the empirical results in the final section examine networks with convolutions and skip connections like ResNets.

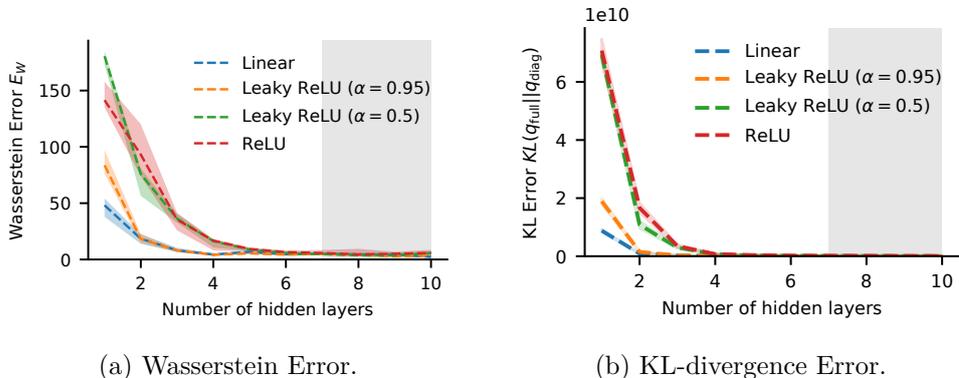


Figure 3.2: For all activations and both error measures, large error in shallow networks almost disappears with depth. All models have $\sim 1,000$ parameters. Shaded depths: HMC samples in these depths show substantially lower test accuracy for the ReLU model ($<95\%$ compared with 99%) so I would not generally depend on them (accuracies for the other models remain good though). See appendix A.1.

the KL-divergence Error is large for shallow networks but rapidly decreases. Although these models are small, this is very direct evidence that there are mean-field modes of the true posterior of a deeper Bayesian neural network. That is, MacKay’s assertion that there are “strong posterior correlations in the parameters” may not be true at all for larger neural networks than the ones he was looking at.

In all cases, this is true regardless of the activation we consider, or whether we use any activation at all. Indeed, we find that a non-linear model with a very shallow non-linearity (LeakyReLU with $\alpha = 0.95$) behaves very much like a deep linear model, while one with a sharper but still shallow non-linearity ($\alpha = 0.5$) behaves much like a ReLU.

3.2 Are There ‘Good’ Approximate Posteriors?

If the true posterior has nearly mean-field modes, it would suggest that a mean-field approximate posterior could be a good approximation to the *parameter* distribution. However, we can make an even more general argument that even when the true posterior is not approximately mean-field, the *predictive* distribution can still be arbitrarily well approximated by a unimodal mean-field approximate posterior, even if the predictive distribution has multiple modes.

We show this using the universal approximation theorem (UAT) due to Leshno et al. [1993] in a stochastic adaptation by Foong et al. [2020]. This shows that a BNN with a mean-field approximate posterior with at least two layers of hidden units can induce a function-space distribution that matches any true posterior distribution over function values arbitrarily closely, given

arbitrary width. Our proof formalizes and extends a remark by Gal [2016, p23] concerning multi-modal posterior predictive distributions.

Proposition 1. *Let $p(\mathbf{y} = \mathbf{Y}|\mathbf{x}, \mathcal{D})$ be the probability density function for the posterior predictive distribution of any given multivariate regression function, with $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{y} \in \mathbb{R}^K$, and \mathbf{Y} the posterior predictive random variable. Let $f(\cdot)$ be a Bayesian neural network with two hidden layers. Let $\hat{\mathbf{Y}}$ be the random vector defined by $f(\mathbf{x})$. Then, for any $\epsilon, \delta > 0$, there exists a set of parameters defining the neural network f such that the absolute value of the difference in probability densities for any point is bounded:*

$$\forall \mathbf{y}, \mathbf{x}, i : \Pr \left(|p(y_i = \hat{Y}_i) - p(y_i = Y_i|\mathbf{x}, \mathcal{D})| > \epsilon \right) < \delta, \quad (3.4)$$

so long as: the activations of f are non-polynomial, non-periodic, and have only zero-measure non-monotonic regions, the first hidden layer has at least $D + 1$ units, the second hidden layer has an arbitrarily large number of units, the cumulative density function of the posterior predictive is continuous in output-space, and the probability density function is continuous and finite non-zero everywhere. Here, the probability bound is with respect to the distribution over a subset of the weights described in the proof, $\boldsymbol{\theta}_{\text{Pr}}$, while one weight distribution $\boldsymbol{\theta}_Z$ remains to induce the random variable $\hat{\mathbf{Y}}$.

The full proof is provided in appendix A.2. Intuitively, we define a $q(\boldsymbol{\theta})$ to induce an arbitrary distribution over hidden units in the first layer and using the remaining weights and hidden layer we approximate the inverse cumulative density function of the true posterior predictive by the UAT. The proof is made much more complicated by the fact that we aim for an architecture which is entirely a stochastic MLP. In fact, all that is required is a single noise variable, Z , and a deterministic single-layer MLP which takes both Z and \mathbf{x} as inputs. Our construction essentially uses the first layer to set up this simpler setting with Z and \mathbf{x} , and then handles the stochasticity of the final layer with a probabilistic generalization of Leshno et al. [1993].

It follows from proposition 1 that there exists a mean-field approximate posterior which induces the true posterior distribution over predictive functions, whether or not the true posterior has an approximately mean-field mode. Our proof strengthens a result by Foong et al. [2020] which considers only the first two moments of the posterior predictive.

There are important limitations to this argument to bear in mind. First, the UAT requires arbitrarily wide models in order to reach arbitrarily precise distribution matching. Second, to achieve arbitrarily small error δ it is necessary to reduce the weight variance. Both of these might result in very low weight-space evidence lower-bounds (ELBOs). Third it may be difficult in practice to choose a prior in weight-space that induces the desired prior in function space. Fourth, although the distribution in weight space that maximizes the marginal likelihood will also maximize the marginal likelihood in

function-space within that model class, the same is not true of the weight-space ELBO and functional ELBO. As a result, it is not entailed that any particular approximation technique, especially variational inference, is likely to find the approximate distributions that would result in a good predictive distribution approximation.

[Foong et al., 2020] also discuss expressivity of BNNs, reaching different conclusions although with no conflict on any proofs or direct results. Here, we consider possible reasons for our differing conclusions.

Empirically, they focus on posterior distributions over function outputs in small models. They find that learned function distributions with MFVI for regression are overconfident. However, their largest experiment uses data with only 16-dimensional inputs, with only 55 training points and very small models with 4 layers of 50 hidden units. In contrast, our work analyses much larger models and datasets. This is where MFVI would be more typical for deep learning but it also makes it harder to compare to a reference posterior in function-space. It is now possible to compare to samples from a reference posterior computed using HMC, for example Wilson et al. [2021].

Theoretically, they show that single-layer mean-field networks cannot have appropriate ‘in-between’ uncertainty, and conjecture that this extends to deeper networks and classification tasks. Their Theorem 1 states that the variance of the function expressed by a single-layer mean-field network between two points cannot be greater than the sum of the variances of the function at those points, subject to a number of very important caveats. The theoretical result applies only to regression models, not to classification. It is also strongest in a 1-dimensional input space. In higher dimensions, they show as a corollary that the ‘in-between’ variance is bounded by the sum of the variances of the hypercube of points including that space, which grows exponentially with dimensionality: even only 10 dimensions, the in-between variance could be 1024 times greater than the average edge variance. The bound is therefore incredibly loose in, for example, computer vision. Last, the result only applies for certain line segments and is sensitive to translation and rotation in input-space.

They also prove their Theorem 3, which establishes that deeper mean-field networks do not have the pathologies that apply in the limited single-layer regression settings identified in Theorem 1. We consider a similar result (proposition 1) which is more general because it considers more than the first two moments of the distribution. Unlike Foong et al. [2020] we see this result as potentially promising that deep mean-field networks can be very expressive.

We note also that we find that deeper networks are able to show in-between uncertainty in regression, if not necessarily capture it as fully as something like HMC. In fig. 3.4 we show how increasing depth increases the ability of MFVI neural networks to capture in-between uncertainty even on low-dimensional data.

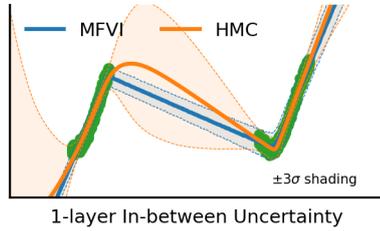


Figure 3.3: 1-layer BNN shows little ‘in-between’ uncertainty.

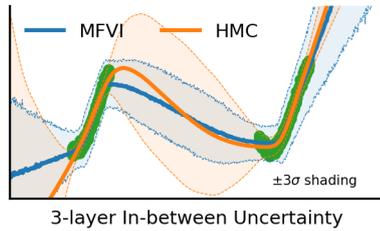


Figure 3.4: 3-layer BNN shows significant ‘in-between’ uncertainty (but not as much as HMC).

Here, each layer has 100 hidden units trained using mean-field variational inference on a synthetic dataset in order to demonstrate the possibility of ‘in-between’ uncertainty. Full experimental settings are provided in table A.1. The toy function used is $y = \sin(4(x-4.3)) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.05^2)$. We sample 750 points in the interval $-2 \leq x \leq -1.4$ and another 750 points in the interval $1.0 \leq x \leq 1.8$. We considered a range of temperatures between 0.1 and 100 in order to select the right balance between prior and data. Note of course that while our figure demonstrates the existence of deep networks that perform well, of course a single case of a one-layer network performing badly does not show that all one-layer networks perform badly.

3.3 Do Approximate Inference Methods Find Good Mean-field Approximations?

Although we have shown empirically that there are nearly-mean-field modes of the true posterior for deep neural networks in parameter-space, and theoretically that mean-field approximate posteriors can be arbitrarily close to the true predictive posterior, we have not yet shown that approximate inference methods succeed in taking advantage of these properties.

Moreover, it is reasonable to hold reservations about the analysis in the previous sections. Although HMC samples are our best lens into the behaviour of the true posterior, it is conceivable that HMC has specific failures in its ability to capture complex correlations that might interfere with our investigations. Similarly, although the universal approximation theorem is powerful, one might wonder if results in wide limits have practical implications for real networks.

In the following sections, we develop a motivating construction showing how depth can allow mean-field networks to induce the same predictive distributions as full-covariance approximate posteriors of shallower networks. We provide a strong correspondence in deep linear models, showing that every matrix-variate Gaussian (MVG) structured covariance approximating distribution can be represented by a three-layer mean-field approximating distribution. We then bridge this correspondence towards neural networks with piecewise-linear activation functions by introducing local product matrices as an analytical tool. These arguments are intended to motivate an understanding of how a deep mean-field model could ‘naturally’ converge on a good solution. This extends on the hypothesis by Hinton and van Camp [1993] that even with a mean-field approximating distribution, during optimization the parameters will find a version of the network where this restriction is least costly. Note, however, that the true posterior over the weights of the shallow network will be different from the true posterior over the weights of the deeper network. Our hypothesis is that, for sufficiently deep networks, both models are sufficiently good (see section 2.2). In this case, finding an approximate posterior for the larger model which is similar in predictive distribution to the smaller model would be sufficient, and is possibly better.

Finally, we examine the question empirically, observing that mean-field approximate posteriors are not obviously worse than structured-covariance approximations in very large models. Moreover, for small models for which full-covariance approximate inference is tractable, the benefits are small and decrease with depth.

Emergent Covariance in Deep Linear Mean-Field Networks

Although we are most interested in neural networks that have non-linear activations, linear neural networks can be analytically useful [Saxe et al., 2014].

Setting the activation function of a neural network, $\phi(\cdot)$, to be the identity turns a neural network into a deep linear model. Without non-linearities the weights of the model just act by matrix multiplication. L weight matrices for a deep linear model can therefore be ‘flattened’ through matrix multiplication into a single weight matrix which we call the *product matrix*— $M^{(L)}$. For a BNN, the weight distributions induce a distribution over the elements of this product matrix. Because the model is linear, there is a one-to-one mapping between distributions induced over elements of this product matrix and the distribution over linear functions $y = M^{(L)}\mathbf{x}$. This offers us a way to examine exactly which sorts of distributions can be induced by a deep linear model on the elements of a product matrix, and therefore on the resulting function-space.

Covariance of the Product Matrix

We derive the analytic form of the covariance of the product matrix in appendix A.2, explicitly finding the covariance of $M^{(2)}$ and $M^{(3)}$ as well as the update rule for increasing L . These results hold for *any* factorized weight distribution with finite first- and second-order moments, not just Gaussian weights. Using these expressions, we show:

Proposition 2. *For $L \geq 3$, the product matrix $M^{(L)}$ of factorized weight matrices can have non-zero covariance between any and all pairs of elements. That is, there exists a set of mean-field weight matrices $\{W^{(l)} | 1 \leq l < L\}$ such that $M^{(L)} = \prod W^{(l)}$ and the covariance between any possible pair of elements of the product matrix:*

$$\text{Cov}(m_{ab}^{(L)}, m_{cd}^{(L)}) \neq 0, \quad (3.5)$$

where $m_{ij}^{(L)}$ are elements of the product matrix in the i^{th} row and j^{th} column, and for any possible indexes a, b, c , and d .

This shows that a deep mean-field linear model is able to induce function-space distributions which would require covariance between weights in a shallower model. We do not show that all possible fully parameterized covariance matrices between elements of the product matrix can be induced in this way.⁵ However, we emphasise that the expressible covariances become very complex. Below, we show that a lower bound on their expressiveness exceeds a commonly used structured-covariance approximate distribution.

⁵E.g., a full-covariance layer has more degrees of freedom than a three-layer mean-field product matrix (one of the weaknesses of full-covariance in practice). An L -layer product matrix of $K \times K$ Gaussian weight matrices has $2LK^2$ parameters, but one full-covariance weight matrix has K^2 mean parameters and $K^2(K^2+1)/2$ covariance parameters. Note also that the distributions over the elements of a product matrix composed of Gaussian layers are not in general Gaussian (see section 3.3 for more discussion of this point).

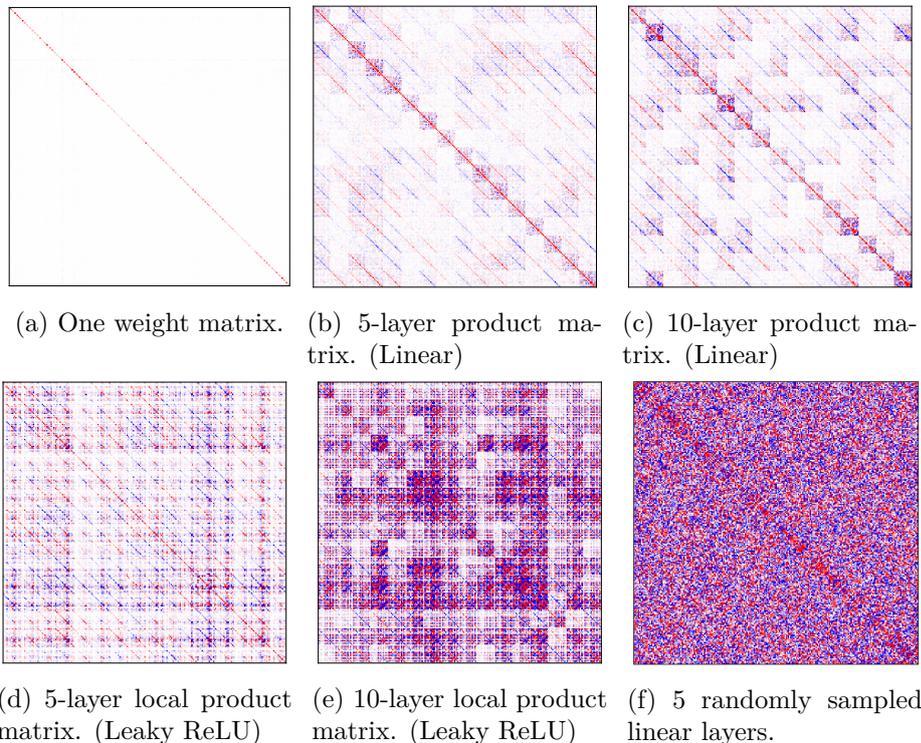


Figure 3.5: Covariance heatmap for mean-field approximate posteriors trained on FashionMNIST. (a) A single layer has diagonal covariance. (b-c) In a deep linear model the *product matrix* composed of L mean-field weight matrices has off-diagonal covariance induced by the mean-field layers. Redder is more positive, bluer more negative. (d-e) For piecewise non-linear activations we introduce ‘local product matrices’ (defined in section 3.3) with similar covariance. Shared activations introduce extra correlations. This lets us extend results from linear to piecewise-linear neural networks. (f) Untrained (randomly sampled) product matrices visually have a noisier and less structured covariance although the mathematical properties of this random matrix are not well understood.

Numerical Simulation

To build intuition, in fig. 3.5a–c we visualize the covariance between entries of the product matrix from a deep mean-field VI linear model trained on FashionMNIST. Even though each weight matrix makes the mean-field assumption, the product develops off-diagonal correlations. The experiment is described in more detail in appendix A.1.

How Expressive is the Product Matrix?

We show that the Matrix Variate Gaussian (MVG) distribution is a special case of the mean-field product matrix distribution. The MVG distribution is used as a structured-covariance approximation by e.g., Louizos and Welling

[2016], Zhang et al. [2018] to approximate the covariance of weight matrices while performing variational inference.⁶ We prove in appendix A.2:

Proposition 3. *The Matrix Variate Gaussian (Kronecker-factored) distribution is a special case of the distribution over elements of the product matrix. In particular, for $M^{(3)} = ABC$, $M^{(3)}$ is distributed as an MVG random variable when A and C are deterministic and B has its elements distributed as fully factorized Gaussians with unit variance.*

Proposition 3 directly entails in the linear case that for any deep linear model with an MVG weight distribution, there exists a deeper linear model with a mean-field weight distribution that induces the same posterior predictive distribution in function-space. This is a **lower bound** on the expressiveness of the product matrix. We have made *very* strong restrictions on the parameterization of the weights for the sake of an interpretable result. The unconstrained expressiveness of the product matrix covariance given in appendix A.2 is much greater. Also note, we do not propose using this, it is purely an analysis tool.

Distribution of the Product Matrix

What is the distribution over elements of the product matrix made of Gaussian layers? It is not Gaussian. For scalars, the product of two independent Gaussian distributions is a generalized χ^2 distribution. The product of N Gaussians with arbitrary mean and variance is unknown outside of special cases (e.g., Springer and Thompson [1970]). An example of a distribution family that *is* closed under multiplication is the log-normal distribution.

Matrix multiplication is important for neural network weights. To preserve distribution through matrix multiplication we would like a distribution which is closed under both addition and multiplication (such as the Generalized Gamma convolution [Bondesson, 2015]) but these are not practical.

However, perhaps even a simple distribution like the Gaussian can maintain roughly similar distributions over product matrix elements as the network becomes deeper. Sometimes, people appeal to the central limit theorem, arguing that the sum of random variables is Gaussian so the distribution is preserved (e.g., [Kingma et al., 2015, Wu et al., 2019]). For only one layer of hidden units, provided K is sufficiently large, this is true. For two or more layers, however, the central limit theorem fails because the elements of the product matrix are no longer

⁶In some settings, MVG distributions can be indicated by the Kronecker-factored or K-FAC approximation. In MVGs, the covariance between elements of an $n_0 \times n_1$ weight matrix can be described as $\Sigma = V \otimes U$ where U and V are positive definite real scale matrices of shape $n_0 \times n_0$ and $n_1 \times n_1$.

independent.

For almost all Gaussians, numerical simulation shows that the resulting product matrix is not remotely Gaussian. However, In fact, although the resulting product matrix is not a Gaussian, we show through numerical simulation that products of matrices with individual weights distributed as $\mathcal{N}(0, 0.23^2)$ have roughly the same distribution over their weights. This, combined with the fact that our choice of Gaussian distributions over weights was somewhat arbitrary in the first place, might reassure us that the increase in depth does not change the model prior in an important way. In fig. 3.6 we plot the probability density function of an arbitrarily chosen entry in the product matrix with varying depths of diagonal Gaussian prior weights. The p.d.f. for 7 layers is approximately the same as the single-layer Gaussian distribution with variance 0.23^2 .

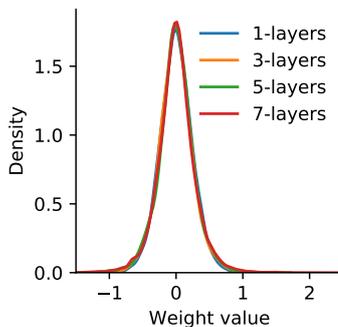


Figure 3.6: Density over arbitrary element of product matrix for L diagonal prior Gaussian weight matrices whose elements are i.i.d. $\mathcal{N}(0, 0.23^2)$. Product matrix elements are not strictly Gaussian, but very close.

Product Matrices in Piecewise-Linear Mean-field BNNs

Neural networks use non-linear activations to increase the flexibility of function approximation. On the face of it, these non-linearities make it impossible to consider product matrices. In this section we show how to define the *local product matrix*, which is an extension of the product matrix to widely used neural networks with piecewise-linear activation functions like ReLUs or Leaky ReLUs. For this we draw inspiration from a proof technique by Shamir et al. [2019] which we extend to stochastic matrices. This analytical tool can be used for any stochastic neural network with piecewise linear activations. Here, we use it to extend proposition 2 to neural networks with piecewise-linear activations.

Multimodal local product matrices

Researchers often critique a Gaussian approximate posterior because it is unimodal *in parameter-space*. We confirm empirically that multiple mean-field layers induce a multi-modal product matrix distribution. Figure 3.7 show a density over an element of the local product matrix from three layers of weights in a Leaky ReLU BNN with $\alpha = 0.1$ from a 3-layer network trained on FashionMNIST.

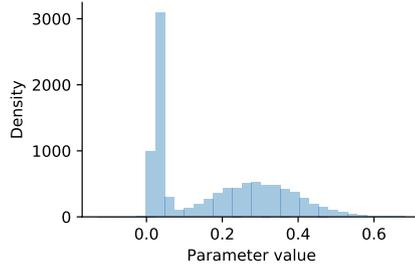


Figure 3.7: Induced product matrices can have multimodal densities

Visually inspecting a sample of 20 elements of this product matrix showed that 12 were multi-modal. We found that without the non-linear activation, none of the product matrix entry distributions examined were multimodal, suggesting that the non-linearities in fact play an important role in inducing rich predictive distributions by creating modes corresponding to activated sign patterns.

Defining a Local Product Matrix

Neural nets with piecewise-linear activations induce piecewise-linear functions. These piecewise-linear neural network functions define hyperplanes which partition the input domain into regions within which the function is linear. Each region can be identified by a sign vector that indicates which activations are ‘switched on’. We show in appendix A.2:

Lemma 1. *Consider an input point $\mathbf{x}^* \in \mathcal{D}$. Consider a realization of the model weights $\boldsymbol{\theta}$. Then, for any \mathbf{x}^* , the neural network function $f_{\boldsymbol{\theta}}$ is linear over some compact set $\mathcal{A}_{\boldsymbol{\theta}} \subset \mathcal{D}$ containing \mathbf{x}^* . Moreover, $\mathcal{A}_{\boldsymbol{\theta}}$ has non-zero measure for almost all \mathbf{x}^* w.r.t. the Lebesgue measure.*

Using a set of N realizations of the weight parameters $\Theta = \{\boldsymbol{\theta}_i \text{ for } 1 \leq i \leq N\}$ we construct a product matrix within $\mathcal{A} = \bigcap_i \mathcal{A}_{\boldsymbol{\theta}_i}$. Since each $f_{\boldsymbol{\theta}_i}$ is linear over \mathcal{A} , the activation function can be replaced by a diagonal matrix which multiplies each row of its ‘input’ by a constant that depends on which activations are ‘switched on’ (e.g., 0 or 1 for a ReLU). This allows us to compute through matrix multiplication a product matrix of L weight layers

$M_{\mathbf{x}^*, \theta_i}^{(L)}$ corresponding to each function realization within \mathcal{A} . We construct a local product matrix random variate $P_{\mathbf{x}^*}$, for a given \mathbf{x}^* , within \mathcal{A} , by sampling these $M_{\mathbf{x}^*, \theta_i}^{(L)}$. The random variate $P_{\mathbf{x}^*}$ is therefore such that y given \mathbf{x}^* has the same distribution as $P_{\mathbf{x}^*} \mathbf{x}^*$ within \mathcal{A} . This distribution can be found empirically at a given input point, and resembles the product matrices from linear settings (see fig. 3.5d–e).

Covariance of the Local Product Matrix

We can examine this local product matrix in order to investigate the covariance between its elements. We prove in appendix A.2 that:

Proposition 4. *Given a mean-field distribution over the weights of neural network f with piecewise linear activations, f can be written in terms of the local product matrix $P_{\mathbf{x}^*}$ within \mathcal{A} .*

For $L \geq 3$, for activation functions which are non-zero everywhere, there exists a set of weight matrices $\{W^{(l)} | 1 \leq l < L\}$ such that all elements of the local product matrix have non-zero off-diagonal covariance:

$$\text{Cov}(p_{ab}^{\mathbf{x}^*}, p_{cd}^{\mathbf{x}^*}) \neq 0, \quad (3.6)$$

where $p_{ij}^{\mathbf{x}^*}$ is the element at the i^{th} row and j^{th} column of $P_{\mathbf{x}^*}$.

Proposition 4 is weaker than the Weight Distribution Hypothesis. Once more, we do not show that all full-covariance weight distributions can be exactly replicated by a deeper factorized network. We now have non-linear networks which give richer functions, potentially allowing richer covariance, but the non-linearities have introduced analytical complications. However, it illustrates the way in which deep factorized networks can emulate rich covariance in a shallower network.

Remark 1. *Proposition 4 is restricted to activations that are non-zero everywhere although we believe that in practice it will hold for activations that can be zero, like ReLU. If the activation can be zero then, for some \mathbf{x}^* , enough activations could be ‘switched off’ such that the effective depth is less than three. This seems unlikely in a trained network, since it amounts to throwing away most of the network’s capacity, but we cannot rule it out theoretically. In appendix A.2 we empirically corroborate that activations are rarely all ‘switched off’ in multiple entire layers.*

Numerical Simulation

We confirm empirically that the local product matrix develops complex off-diagonal correlations using a neural network with Leaky ReLU activations trained on FashionMNIST using mean-field variational inference. We estimate

the covariance matrix using 10,000 samples of a trained model (fig. 3.5d–e). Just like in the linear case (fig. 3.5a–c), as the model gets deeper the induced distribution on the product matrix shows complex off-diagonal covariance. There are additional correlations between elements of the product matrix based on which activation pattern is predominantly present at that point in input-space. See appendix A.1 for further experimental details.

Does Mean-field Approximate Inference Work Empirically?

Here, we compare the performance of Bayesian neural networks with complex posterior approximations to those with mean-field approximations. We show that over a spectrum of model sizes, examining both variational inference and amortized stochastic gradient Markov chain Monte Carlo, performance does not seem to be greatly determined by the approximation.

Depth in Full- and Diagonal-covariance Variational Inference. Training with full-covariance variational inference is intractable, except for very small models, because of optimization difficulties. In fig. 3.8, we show the test cross-entropy of small models of varying depths on the Iris dataset from the UCI repository. With one layer of hidden units the full-covariance posterior achieves lower cross-entropy. For deeper models, however, the mean field network matches the full-covariance one. Full details of the experiment can be found in appendix A.1.

Structured- and Diagonal-covariance Uncertainty on CIFAR-100. Although we cannot compute samples from the true posterior in larger models, we attempt an approximate investigation using SWAG [Maddox et al., 2019]. This involves fitting a Gaussian distribution to approximate SG-MCMC samples on CIFAR-100. SWAG approximates the Gaussian distribution with a low rank empirical covariance matrix, while SWAG-Diag uses a factorized Gaussian. We show in fig. 3.9 that there is no observable difference in negative log-likelihood or accuracy between the diagonal and low-rank approximation. All of the models considered have more than two layers of hidden units (the minimum size of a PresNet). This suggests that there is a mode of the true posterior over weights for these deeper models that is sufficiently mean-field that a structured approximation provides little or no benefit. It also suggests that past a threshold of two hidden layers, further depth is not essential.

Large-model Mean-field Approximations on Imagenet. The performance of mean-field and structured-covariance methods on large-scale tasks can give some sense of how restrictive the mean-field approximation is. Mean-field methods have been shown to perform comparably to structured methods in large scale settings like Imagenet, both in accuracy and measures of uncertainty like log-likelihood and expected calibration error (ECE) (see table 3.2).

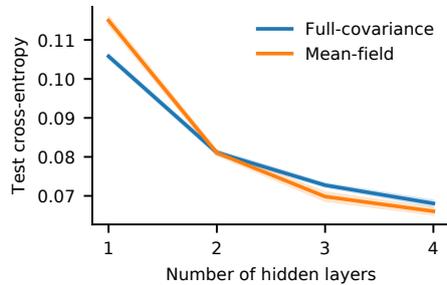


Figure 3.8: Full- vs diagonal-covariance. After two hidden layers mean-field matches full-covariance. Iris dataset.

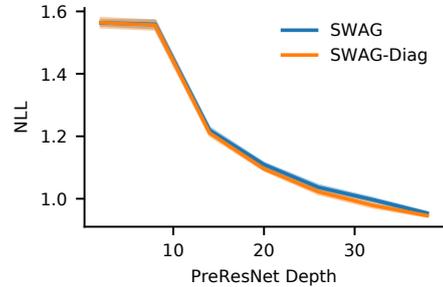


Figure 3.9: CIFAR-100. Diagonal- and Structured-SWAG show similar log-likelihood in PresNets of varying depth.

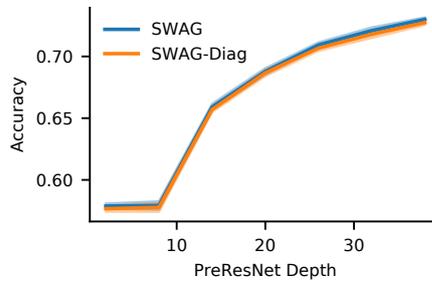


Figure 3.10: CIFAR-100. Accuracy for diagonal and low-rank covariance SWAG. Like log-likelihood, there is no clear difference in performance between these models, all of which are above the depth threshold implied by our work.

Architecture	Method	Covariance	Accuracy	NLL	ECE
ResNet-18	VOGN [‡]	Diagonal	67.4% ± 0.263	1.37 ± 0.010	0.029 ± 0.001
ResNet-18	Noisy K-FAC ^{††}	MVG	66.4% ± n.d.	1.44 ± n.d.	0.080 ± n.d.
DenseNet-161	SWAG-Diag [†]	Diagonal	78.6% ± 0.000	0.86 ± 0.000	0.046 ± 0.000
DenseNet-161	SWAG [†]	Low-rank + Diag	78.6% ± 0.000	0.83 ± 0.000	0.020 ± 0.000
ResNet-152	SWAG-Diag [†]	Diagonal	80.0% ± 0.000	0.86 ± 0.000	0.057 ± 0.000
ResNet-152	SWAG [†]	Low-rank + Diag	79.1% ± 0.000	0.82 ± 0.000	0.028 ± 0.000

Table 3.2: Imagenet. Comparison of diagonal-covariance/mean-field (in grey) and structured-covariance methods on Imagenet. The differences on a given architecture between comparable methods is small and has inconsistent sign. [†] [Maddox et al., 2019]. [‡] [Osawa et al., 2019]. ^{††} [Zhang et al., 2018] as reported by Osawa et al. [2019].

Architecture	Method	Covariance	Accuracy	NLL	ECE
VGG-16	SWAG-Diag [†]	Diagonal	93.7% ± 0.15	0.220 ± 0.008	0.027 ± 0.003
VGG-16	SWAG [†]	Low-rank + Diag	93.6% ± 0.10	0.202 ± 0.003	0.016 ± 0.003
VGG-16	Noisy Adam ^{‡‡}	Diagonal	88.2% ± n.d.	n.d.	n.d.
VGG-16	BBB ^{‡‡}	Diagonal	88.3% ± n.d.	n.d.	n.d.
VGG-16	Noisy KFAC ^{‡‡}	MVG	89.4% ± n.d.	n.d.	n.d.
PreResNet-164	SWAG-Diag [†]	Diagonal	96.0% ± 0.10	0.125 ± 0.003	0.008 ± 0.001
PreResNet-164	SWAG [†]	Low-rank + Diag	96.0% ± 0.02	0.123 ± 0.002	0.005 ± 0.000
WideResNet28x10	SWAG-Diag [†]	Diagonal	96.4% ± 0.08	0.108 ± 0.001	0.005 ± 0.001
WideResNet28x10	SWAG [†]	Low-rank + Diag	96.3% ± 0.08	0.112 ± 0.001	0.009 ± 0.001
ResNet-18	VOGN [‡]	Diagonal	84.3% ± 0.20	0.477 ± 0.006	0.040 ± 0.002
AlexNet	VOGN [‡]	Diagonal	75.5% ± 0.48	0.703 ± 0.006	0.016 ± 0.001

Table 3.3: CIFAR-10. For a given architecture, it does not seem that mean-field (grey) methods systematically perform worse than methods with structured covariance, although there is some difference in the results reported by different authors. [†] [Maddox et al., 2019]. [‡] [Osawa et al., 2019]. ^{‡‡} [Zhang et al., 2018].

For VOGN [Osawa et al., 2019] which explicitly optimizes for a mean-field variational posterior, the mean-field model is marginally better in all three measures. For SWAG, the accuracy is marginally better and log-likelihood and ECE marginally worse for the diagonal approximation. This is consistent with the idea that there are some modes of large models that are approximately mean-field (which VOGN searches for but SWAG does not) but that not all modes are. These findings offer some evidence that the importance of structured covariance is at least greatly diminished in large-scale models, and may not be worth the additional computational expense and modelling complexity. ⁷

A similar evaluation for CIFAR-10 shows a similar lack of clear advantage to using a richer covariance for deeper models table 3.3. Within the same architecture, there is little evidence of systematic differences between mean-field and structured-covariance methods and any differences which do appear are marginal. Note that Zhang et al. [2018] report difficulty applying batch normalization to mean-field methods, but Osawa et al. [2019] report no difficulties applying batch normalization for their mean-field variant of Noisy Adam. For this reason, we report the version of Noisy KFAC run without batch normalization to make it comparable with the results shown for Bayes-by-Backprop (BBB) and Noisy Adam. With batch normalization, Noisy KFAC gains some accuracy, reaching 92.0%, but this seems to be because of the additional regularization, not a property of the approximate posterior family.

⁷The standard deviations, of course, underestimate the true variability of the method in question on Imagenet as they only consider difference in random seed with the training configuration otherwise identical. Fuller descriptions of the experimental settings used by the authors are provided in the cited papers.

3.4 Discussion

Researchers have a longstanding intuition that the mean-field approximation for Bayesian neural networks is a severe one. However, this impression has not been based on a rigorous notion of severity because of the difficulty of understanding what makes a ‘good’ approximate posterior in the first place. In this chapter, we have examined three underlying questions:

- Does the *true* posterior distribution have strong correlations between the parameters?
- Do there exist approximate posterior distributions that are ‘good’ even if they are mean-field?
- Do actual methods for approximate Bayesian inference uncover these ‘good’ approximations?

For the first, we have shown the way in which the *architecture* of the parametric model influences characteristics of the *true* posterior. This has an important and underappreciated significance: good approximate Bayesian inference depends heavily on *matching* the architecture to the approximation scheme and dataset because different architectures have different posteriors on different datasets which will sometimes suit some approximation schemes better than others. In particular, we have argued that mean-field approximating distributions are more likely to be a good fit in deep neural networks than shallow ones.

For the second, we have proven under mild assumptions that there exist approximating distributions over Bayesian neural networks of sufficient depth and width which are arbitrarily close to the true posterior predictive.

For the third, we have provided empirical evidence that schemes like variational inference and stochastic gradient Markov chain Monte Carlo do in fact uncover these modes. We have provided a motivating construction for how this might happen because of the ability of blocks of several diagonal-covariance layers to ‘simulate’ shallower layers with full-covariance.

Despite this, it remains the case that mean-field variational inference is rarely used in practice because of poor results. In the next chapter, we consider ways in which the approximating distribution interacts not just with the data and architecture but also the *optimization process*. We will show how the standard Gaussian approximating posterior has bad sampling properties that affect optimization, and provide an alternative distribution which fixes these properties.

Chapter 4

Approximation Assumptions Affect Optimization

Approximate Bayesian inference is often performed using optimization techniques. This is most marked for variational inference (VI) which converts the inference problem into an optimization problem on the evidence lower bound (ELBO).¹ A topic which has received little attention is the ways in which the choice of approximating distribution interact with the procedure used to optimize the ELBO. In this chapter, we show how the standard multivariate Gaussian approximating distribution has pathological high-dimensional properties: typical points are far from both each other and the mean, resulting in high variance estimates. As a result, we propose an alternative sampling distribution with more desirable sampling properties. We show how this reduces gradient variance and can lead to ‘better’ approximate posteriors. This is an example of the ways in which careful use of approximations can influence the performance of approximate Bayesian inference in ways that are not connected to the quality of the underlying probabilistic model.

In the previous chapter, we argued that the mean-field approximation in approximate Bayesian inference for sufficiently deep neural networks is not as severe an approximation as people often assume. Nevertheless, researchers have difficulties in employing mean-field methods in practice. To make mean-field variational inference (MFVI) work, researchers often resort to ad-hoc tweaks to the loss or optimization process which side-step the variational inference arguments that motivate the approach in the first place! Our analysis

¹Stochastic gradient Markov chain Monte Carlo methods also use optimization methods, but in a very different way. It is possible that the approach considered in this chapter would influence the mixing properties of SG-MCMC, but it is out of scope for this work.

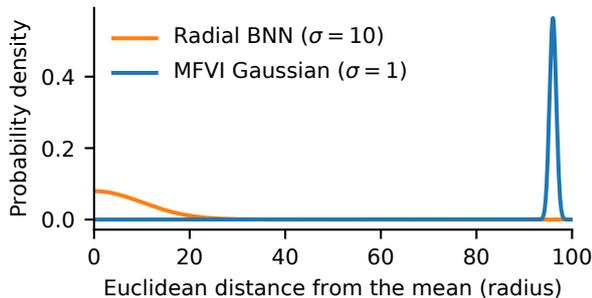


Figure 4.1: MFVI uses a multivariate Gaussian approximate posterior whose probability mass is tightly clustered at a fixed radius from the mean depending on the number of dimensions—the ‘soap-bubble’. In our Radial BNN, samples from the approximate posterior are more reflective of the mean. This helps training by reducing gradient variance. The plot shows the p.d.f. based on dimensionality of a 3×3 convolutional layer with 64 channels.

of optimization helps us understand why MFVI struggles despite the fact that the approximating distribution is capable of expressing sufficiently rich predictive distributions.

4.1 Understanding the Soap Bubble

The ‘soap-bubble’ is a well-known property in multi-variate Gaussian distributions as the number of dimensions increases (e.g., see Bishop [2006], Betancourt [2018]). Although the highest probability density is near the mean, because there is just so much more volume further from the mean in high-dimensional spaces most of the probability mass is located in a narrow band—a ‘soap-bubble’. This band is where the opposing pressures of growing space and shrinking density intersect. For high-dimensional spaces, the band is both vanishingly narrow and far away from the mean.

One way to understand this is to examine the probability density function of the multivariate Gaussian along its radius (following the derivation of [Bishop, 2006]). Consider a D -dimensional isotropic Gaussian from which we sample $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2)$. Take a thin shell with thickness η at radius r away from the mean $\boldsymbol{\mu}$. As η tends to zero, the probability density function over the radius is given by:

$$\lim_{\eta \rightarrow 0} p(r - \eta < \|\mathbf{w} - \boldsymbol{\mu}\| < r + \eta) \quad (4.1)$$

$$= \underbrace{\frac{S_D}{(2\pi\sigma^2)^{D/2}}}_{\text{Normalizing constant}} \cdot \underbrace{r^{D-1}}_{\text{Growing volume}} \cdot \underbrace{e^{-\frac{r^2}{2\sigma^2}}}_{\text{Shrinking density}} \quad (4.2)$$

where S_D is the surface area of a hypersphere in a D -dimensional space.

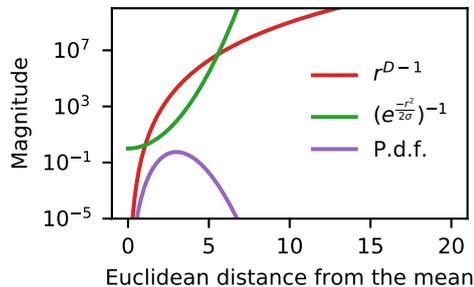


Figure 4.2: We can understand the ‘soap-bubble’ by looking at components of the p.d.f. in eq. 4.2. For small r the volume term (red) dominates and the normalized p.d.f. is very small. For big r the Gaussian density term (inverse shown in green) dominates and the p.d.f. is small again. Almost all the probability mass is in the intermediate region where neither term dominates: the ‘soap bubble’. The intermediate region becomes narrower and further from the mean as D is bigger. Here we show $D = 10$.

The two non-normalization terms are shown in fig. 4.2. The inverted density (in green) grows quadratically on a log-scale while the volume (in red) logarithmically. To begin, the low volume dominates and the p.d.f. over the radius is small. At the end, the low density dominates and the p.d.f. is small again. Most of the mass is in a band where the two overlap—and the position and narrowness of this band depends on the dimensionality of \mathbf{w} .

The Soap Bubble In Variational Inference

A powerful way to optimize the variational posterior for a neural network is using Monte Carlo expectations to compute the ELBO [Graves, 2011, Blundell et al., 2015]. The ELBO for a variational approximate posterior distribution $q(\mathbf{w})$ over the weights of our neural network \mathbf{w} relative to a true posterior distribution $p(\mathbf{w} | \mathcal{D})$ can be written

$$\text{ELBO}(q, p) = \mathbb{E}_{q(\mathbf{w})} [\log p(\mathcal{D} | \mathbf{w})] - D_{\text{KL}}(q(\mathbf{w}) \| p(\mathbf{w})) \quad (4.3)$$

$$= \mathbb{E}_{q(\mathbf{w})} [\log p(\mathcal{D} | \mathbf{w})] - \mathbb{E}_{q(\mathbf{w})} \left[\log \frac{q(\mathbf{w})}{p(\mathbf{w})} \right]. \quad (4.4)$$

Estimating the ELBO, therefore, involves estimating two expectations over the variational distribution. The variance of Monte Carlo expectations is a significant problem which has received much attention, for example through the use of importance sampling [Kahn and Marshall, 1953]. Of fundamental importance, however, are the sampling properties of $q(\mathbf{w})$.

The soap-bubble has the consequence that almost all samples from the distribution are very distant from the mean. Typical samples from \mathbf{w} are a distance from the mean proportional to $\sigma\sqrt{D}$, for standard deviation parame-

ter σ and the number of parameters D .² This effect is therefore more extreme in larger neural networks.

In addition, because of the geometry of hyper-spheres in high-dimensional spaces, with high probability each sample is far from all the other samples. Similarly, this distance is proportional to $\sigma\sqrt{D}$ and the probability distribution of distances becomes increasingly tight as D grows.

All else equal, we might expect this to lead to predictions and losses from multiple samples of the weights which are less correlated with each other than if the samples were near to each other in weight-space. We anticipate (and demonstrate in §4.4) that the distance between samples from the MFVI approximate posterior makes the gradient estimator of the log-likelihood term of the loss in eq. (4.9) have a large variance, which makes optimization more difficult.

Soap bubbles in related settings

In Bayesian optimization, Oh et al. [2018] consider the difficulties posed by soap-bubbles. Oh et al. [2019] use hyperspherical coordinate system for variational inference in a way that removes the soap-bubble. They motivate their posterior as a way to model correlations between weights to remove the restrictiveness of the mean-field assumption (c.f., 3). As a result, although their approach is superficially similar to ours in some ways, their approximate posterior is radial over each row in their weight matrix, rather than the whole layer, and they introduce an expensive posterior distribution (von Mises-Fisher) to explicitly model weight correlations within rows, whereas we do *not* seek to learn any correlations between parameters in the hyperspherical space. That is, their method uses a different technique to solve a different problem.

4.2 Overcoming the Soap Bubble

To address this, we propose an alternative approximate posterior distribution without a ‘soap-bubble’. Our strategy is to use a simple approximate posterior distribution in a hyperspherical space corresponding to each layer, which has been chosen to avoid a soap-bubble, and then transform this distribution into the coordinate system of the weights. We show that the Radial BNN can be sampled efficiently in weight-space, without needing explicit coordinate transformations, and derive an analytic expression for the loss that makes training as fast and as easy to implement as MFVI.

²For simplicity, to calculate the distance of typical samples from the mean we imagine an isotropic posterior. The approximate posteriors are not isotropic, which means the true expression is slightly more complicated, but the pattern is similar.

A ‘soap-bubble’ arises when, for large D , the probability density function over the radius from the mean is sharply peaked at a large distance from the mean (see fig. 4.1). Therefore, we pick a probability distribution which cannot have this property. We can easily write down a probability density function which cannot have a ‘soap-bubble’ by explicitly modelling the radius from the mean. The hyperspherical coordinate system suits our needs: the first dimension is the radius and the remaining dimensions are angles. We pick the simplest practical distribution in hyperspherical coordinates with no soap bubble:

- In the radial dimension: $r = |\tilde{r}|$ for $\tilde{r} \sim \mathcal{N}(0, 1)$.³
- In the angular dimensions: uniform distribution over the hypersphere—all directions equally likely.

A critical property is that it is easy to sample this distribution *in the weight-space coordinate system*—we wish to avoid the expense of explicit coordinate transformations when sampling from the approximate posterior. Instead of sampling the posterior distribution directly, we use the local reparameterization trick Rezende et al. [2014], Kingma et al. [2014], and sample the noise distribution instead. This is similar to Graves [2011] and Blundell et al. [2015] who sample their weights

$$\mathbf{w} := \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_{\text{MFVI}}, \quad (4.5)$$

where $\boldsymbol{\epsilon}_{\text{MFVI}} \sim \mathcal{N}(0, \mathbf{I})$ and \cdot is an element-wise multiplication. In order to sample from the Radial BNN posterior we make a small modification:

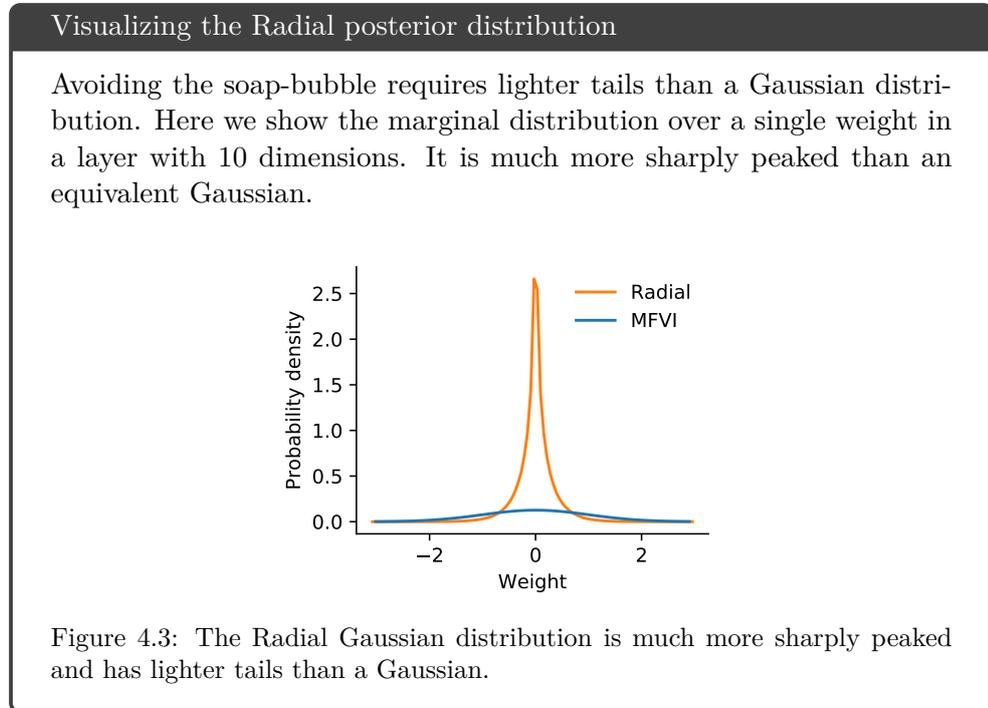
$$\mathbf{w}_{\text{radial}} := \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \frac{\boldsymbol{\epsilon}_{\text{MFVI}}}{\|\boldsymbol{\epsilon}_{\text{MFVI}}\|} \cdot r, \quad (4.6)$$

which works because dividing a multi-variate Gaussian random variable by its norm provides samples from a direction uniformly selected from the unit hypersphere [Muller, 1959, Marsaglia, 1972]. As a result, sampling from our posterior is nearly as cheap as sampling from the MFVI posterior. The only extra steps are to normalize the noise, and multiply by a scalar Gaussian random variable.

In fact, we can speed this up even further by noting that in high-dimensional space the norm $\|\boldsymbol{\epsilon}_{\text{MFVI}}\|$ can be calculated analytically with high probability (precisely because of the soap-bubble!). This saves us the step of normalizing the noise, making each additional sample from the approximate posterior only more expensive than the standard MFVI samples by scalar multiplication and division.

³The absolute value rules out negative radii. In practice, we can neglect the absolute value because of the rotational symmetry in angular dimensions.

In our implementations we sample from the Radial distribution separately for each layer, and indeed we sample weights and biases separately.



Evaluating the Objective

To use our approximate posterior for variational inference we must be able to estimate the ELBO loss. The Radial BNN posterior does not change how the expected log-likelihood is estimated, using mini-batches of data points and MC integration.

The KL divergence between the approximate posterior and prior can be written:

$$\begin{aligned}
 D_{\text{KL}}(q(\mathbf{w}) \parallel p(\mathbf{w})) &= \int q(\mathbf{w}) \log[q(\mathbf{w})] d\mathbf{w} - \int q(\mathbf{w}) \log[p(\mathbf{w})] d\mathbf{w} \\
 &= \mathcal{L}_{\text{entropy}} - \mathcal{L}_{\text{cross-entropy}}.
 \end{aligned}
 \tag{4.7}$$

We estimate the cross-entropy term using MC integration, just by taking samples from the posterior and averaging their log probability under the prior. We find that this is low-variance in practice, and is often done for MFVI as well [Blundell et al., 2015].

We can evaluate the entropy of the posterior analytically. We derive the entropy term in Appendix B.1:

$$\mathcal{L}_{\text{entropy}} = - \sum_i \log[\sigma_i] + \text{const}.
 \tag{4.8}$$

where i sums over the weights. This is, up to a constant, the same as when using an ordinary multivariate Gaussian in MFVI.⁴ Although they are not needed for optimization, for sake of completeness, we also derive the constant terms in the Appendix.

In Appendix B.2, we also provide a derivation of the cross-entropy loss term in the case where the prior is a Radial BNN. This is useful in continual learning (see section 5.1) where we use the posterior from training one model as a prior when training another but is more computationally expensive than using a Gaussian prior because it requires an explicit change of variables.

Code implementing Radial BNNs can be found at https://github.com/SebFar/radial_bnn. Training Radial BNNs has the same computational complexity as MFVI— $\mathcal{O}(D)$, where D is the number of weights in the model.

4.3 Diabetic Retinopathy Pre-screening

Here, we establish the robustness and performance of Radial BNNs using a Bayesian medical imaging task identifying diabetic retinopathy in ‘fundus’ eye images [Leibig et al., 2017], using models with ~ 15 M parameters and inputs with $\sim 260,000$ dimensions, in section 4.3 (see fig. 4.4). In later chapters, we return to the Radial BNN posterior when demonstrating evaluations based on active and continual learning.

In our experiments, Radial BNNs are more robust to hyperparameter choice than MFVI and that Radial BNNs outperform the current state-of-the-art Monte-Carlo (MC) dropout and deep ensembles on this task.⁵ Our work is focused on large datasets and big models, which is where the most exciting application for deep learning are. That is where complicated variational inference methods that try to learn weight covariances become intractable, and

⁴In the common case of a unit multivariate Gaussian prior and an approximate posterior $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ over the weights \mathbf{w} , the negative evidence lower bound (ELBO) objective is:

$$\mathcal{L}_{\text{MFVI}} = \underbrace{\sum_i \frac{1}{2} [\sigma_i^2 + \mu_i^2]}_{\text{prior cross-entropy}} - \underbrace{\sum_i \log[\sigma_i]}_{\text{approximate-posterior entropy}} - \underbrace{\mathbb{E}_{\mathbf{w} \sim q(\mathbf{w})} [\log p(\mathbf{y}|\mathbf{w}, \mathbf{X})]}_{\text{data likelihood}}. \quad (4.9)$$

⁵In some recent work including Nado et al. [2021] Radial BNNs perform worse than MFVI or deep ensemble methods. There are several possible explanations. First, many implementations use a variety of implicit ‘hacks’ for MFVI that have evolved through countless hours of researcher time. It may be that similar tuning would let Radial BNNs to reach higher levels of performance. Second, our experiments might overestimate the performance of Radial BNNs. Although we made every effort to tune the hyperparameters of all models using the same number of seeds and Bayesian optimization, I wanted my method to perform better and this may have subtly influenced experimental decisions. Third, details of the experiments and architecture varied between settings. For example, we used multiple variational samples in the forward pass and pretrained the means using maximum likelihood loss for several epochs before performing variational inference, and used VGG-style architectures.

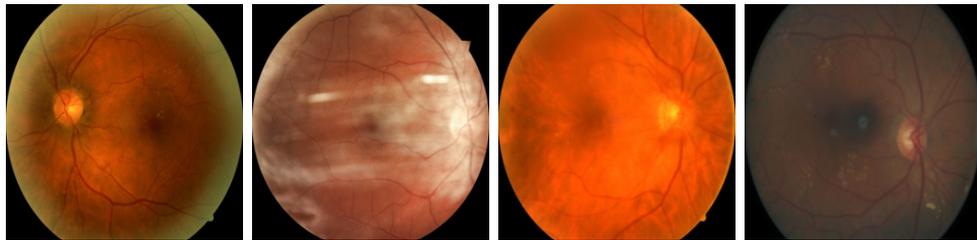


Figure 4.4: Examples from retinopathy dataset. In order: healthy eye; healthy eye with camera artefacts;diseased eye; diseased eye. The chance that bad images cause misdiagnosis makes uncertainty-aware models vital.

where the ‘soap-bubble’ pathology emerges. We show that on a large-scale diabetic retinopathy diagnosis image classification task: our radial posterior is more accurate, has better calibrated uncertainty, and is more robust to hyper-parameters than MFVI with a multivariate Gaussian and therefore requires fewer iterations and less experimenter time.

Dataset description

We perform classification on a dataset of ‘fundus’ images taken of the back of retinas in order to diagnose diabetic retinopathy [Kaggle, 2015] building on Leibig et al. [2017] and Filos et al. [2019].⁶ Diabetic retinopathy is graded in five stages, where 0 is healthy and 4 is the worst. Following Leibig et al. [2017], we distinguish the healthy (classes 0 and 1) from those that require medical observation and attention (2, 3, and 4). Images (512x512) include left and right eyes separately, which are not considered as a pair by the models, and come from two different camera technologies in many different physical locations. Model uncertainty is used to identify badly-taken or confusing images which could be used to refer affected patients to experts for more detailed examination.

The diabetic retinopathy dataset has several properties which differ from standard image classification datasets but are similar to many real-world settings. First, there is substantial class imbalance, with many more healthy than unhealthy images. Second, the consequences of misclassification are much greater in the case of a false negative (a sick person does not receive treatment) than a false positive (a healthy person receives an unnecessary but non-invasive follow-up examination). Third, the dataset contains a mixture of left- and right- eyes which appear slightly differently as well as images taken using two different camera orientations (coded using a notch at the top and

⁶The precise dataset and augmentations which we used came from an early version of Filos et al. [2019], which is now different from the publicly available dataset. We describe the data augmentations, which differ from the current benchmark, in detail in appendix B.3 so that the experiments could be reproduced if desired.

bottom right of the image). These make data-augmentation using reflective symmetries particularly important for training.

Performance and Calibration

In table 4.1 we compare the classification area under the curve (AUC) of the receiver operating characteristic of predicted classes (higher is better).⁷ We consider the model performance under different thresholds for referring data to experts. At 0%, the model makes predictions about all data. At 30%, the 30% of images about which the model is least confident are referred to experts and do not get scored for the model—the AUC should therefore become higher if the uncertainties are well-calibrated. We show that our Radial BNN outperforms MFVI by a wide margin, and even outperforms MC dropout. While the deep ensemble is better at estimating uncertainty than a single Radial BNN, it has three times as many parameters. An ensemble of Radial BNNs outperforms deep ensembles at all levels of uncertainty. Radial BNN models trained on this dataset also show empirical calibration that is closer to optimal than other methods (see fig. 4.5b).

The model hyperparameters were all selected individually by Bayesian optimization using ten runs. Full hyperparameters and search strategy, pre-processing, and architecture are provided in appendix B.3. We include both the original MC dropout results from Leibig et al. [2017] as well as our reimplementation using the same model architecture as our Radial BNN model. The only difference between the MC dropout and Radial BNN/MFVI architectures is that we use more channels for MC dropout, so that the number of parameters is the same in all models. We pick the VGG-16 architecture for comparability with prior work, but note that Nado et al. [2021] evaluate performance on a ResNet. As discussed above, they find that a method similar to our ‘MFVI w/ tweaks’ outperforms Radial BNNs, which might mean that the behaviours described in section 4.4 are alleviated by the presence of residual connections. We estimate the standard error of the AUC using bootstrapping.

The difficulty of ‘naive’ mean-field VI

In practice, training BNNs with MFVI is difficult and people adopt unprincipled ‘hacks’. For example, in order to produce variational inference methods that work on larger datasets and architectures, [Nado et al., 2021] use a number of ‘hacks’ including adjusting the ‘prior’ to

⁷We follow prior work which selected AUC as a metric because classes are unbalanced (mostly healthy) which makes the AUC more reflective of model performance than something like accuracy. Plausibly better would be to construct a loss which takes the relative value of different kinds of misclassification into account.

always have the same mean as the current approximate posterior and scaling the KL-divergence term of the loss between 0.5% and 45% of its true value during training. Indeed, because they use learning-rate decay schedules, even if they had increased the KL-divergence scaling term all the way to 1 by the end of training, one could not expect that the approximate posterior was even close to the optimum of the ELBO. As a result, their implementation is far from performing principled variational inference. They are not an outlier in this regard, but are selected only because they have a prominent, public, and well-documented code-base. These tricks are widely used. Some further tricks include performing early-stopping alongside a KL-annealing schedule or a tiny initialized variance (e.g., Nguyen et al. [2018]) or simply performing ‘variational inference’ using the log-likelihood loss only [Fortunato et al., 2018].

Others have attempted to scale mean-field variational inference to larger settings, such as Osawa et al. [2019] who use variational online Gauss-Newton methods [Khan et al., 2018]. However, this method relies on several significant approximations to make estimates of the Hessian tractable and the performance of the method lags significantly behind deep ensembles.

From an alternative perspective, Wu et al. [2019] argue that mean-field VI is sensitive to initialization and priors. They see the variance of ELBO estimates as the problem and introduce a deterministic alternative. We agree that this is a crucial problem, but offer a simpler and cheaper alternative solution which does not require extra assumptions about the distribution of activations.

Robustness

The radial posterior was more robust to hyperparameter variation (Figure 4.5a). We assess robustness on a downsampled version of the diabetic retinopathy dataset (256x256) using a smaller model with a similar architecture to VGG-16, but which trained to convergence in about a tenth the time and had only $\sim 1.3\text{M}$ parameters. We randomly selected 86 different runs from plausible optimizer, learning rate, learning rate decay, batch size, number of variational samples per forward pass, and initial variance. 82% of hyperparameters tried for the MFVI baseline resulted in barely any improvement over randomly guessing, compared with 39% for the radial posterior. 44% of configurations for our radial posterior reached good AUCs, compared with only 11% for MFVI. This is despite the fact that we did allow models to pre-train the means using a negative log-likelihood loss for one epoch before beginning ELBO training, a common tweak to improve MFVI.

Method	Architecture	# Params	Epoch Train Time (m)	ROC-AUC for different percent data referred to experts			
				0%	10%	20%	30%
MC-dropout	[Leibig et al., 2017]	~21M	-	92.7±0.3%	93.8±0.3%	94.7±0.3%	95.6±0.3%
MC-dropout	VGG-16	~15M	5.6	93.0±0.04%	94.1±0.05%	94.5±0.05%	95.1±0.07%
MFVI	VGG-16*	~15M	16.0	63.6±0.13%	63.5±0.09%	63.5±0.09%	62.6±0.10%
MFVI w/ tweaks	VGG-16*	~15M	16.0	93.9±0.04%	94.4±0.05%	95.4±0.04%	96.4±0.05%
Radial BNN	VGG-16*	~15M	16.2	94.3±0.04%	95.3±0.06%	96.1±0.06%	96.8±0.04%
Deep Ensemble	3xVGG-16	~45M	16.8†	93.9±0.04%	96.0±0.05%	96.6±0.04%	97.2±0.04%
Radial Ensemble	3xVGG-16*	~45M	48.6†	94.5±0.05%	97.9±0.04%	98.0±0.03%	98.1±0.03%

Table 4.1: Diabetic Retinopathy Prescreening: Our Radial BNN outperforms SOTA MC-dropout and is able to scale to model sizes that MFVI cannot handle without ad-hoc tweaks (see section 4.3). Even with tweaks, Radial BNN still outperforms. Deep Ensembles outperform a single Radial BNN at estimating uncertainty, but are worse than an ensemble of Radial BNNs with the same number of parameters. \pm indicates bootstrapped standard error from 100 resamples of the test data. We use an adapted VGG-16* model for our Bayesian deep learning methods which have fewer channels so that # of parameters is the same as non-Bayesian model. This architecture was chosen to match prior work [Leibig et al., 2017, Filos et al., 2019]. Nado et al. [2021] compare Radial BNNs to other variational methods on residual networks rather than the VGG architecture, and find the performance of their other baselines (most comparable to MFVI w/ tweaks in design) is slightly better. This may be because residual connections reduce the effects we observed in this paper. † three times the train time of a single model, though this could be parallelized.

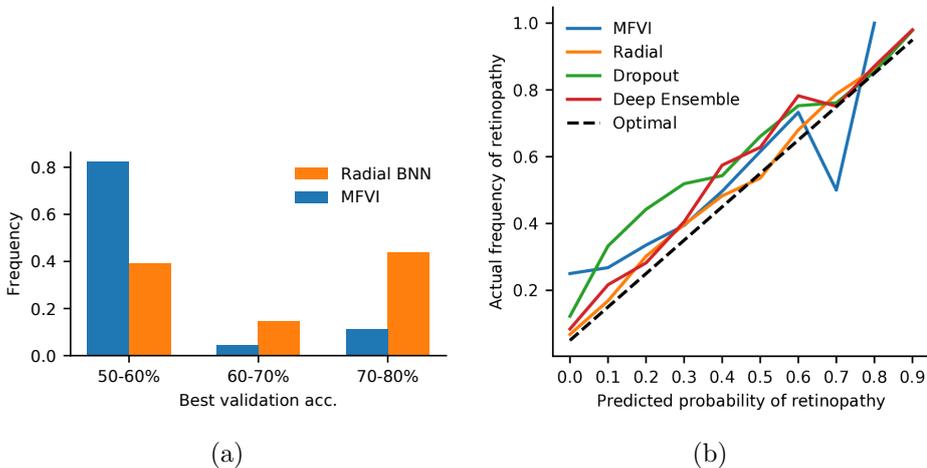


Figure 4.5: (a) The Radial BNN posterior is more robust to hyperparameters on a downsampled version of the retinopathy dataset. Over 80% of configurations for the MFVI baseline learned almost nothing. 4 times more Radial BNNs had good accuracies than MFVI models. (b) Radial BNN is almost perfectly calibrated, compared with MC dropout and deep ensembles (overconfident) and ordinary MFVI without ad-hoc tweaks which is not well calibrated. X-axis labels are the lower-bound of each range (e.g., 0.0 is 0.0-0.1).

UCI experiments and Bayesian deep learning

These retinopathy experiments are orders of magnitude larger than most other VI work at the time, which was heavily influenced by experiments on UCI datasets from Hernández-Lobato and Adams [2015] with between 4 and 16 input dimensions and using fewer than 2000 parameters.

Radial BNNs, like MFVI, perform worse than more expensive methods on the UCI datasets. This is expected—our method is designed for models with high-dimensional weight-space, not for the artificial constraints of the experimental settings used on the UCI evaluations. An over-reliance on UCI focuses on settings where Bayesian deep learning is not particularly suitable because the advantages of neural networks are not critical. Recently, researchers have moved towards more large-scale evaluations for uncertainty.

Nevertheless, completely abandoning UCI may also be a mistake, because it is important to understand the shortcomings that Bayesian deep learning can show on regression problems. Most recent Bayesian deep learning work (including this thesis) overly emphasises computer vision classification. These are interesting, important, and a relevant application where deep learning has been phenomenally successful. Nevertheless, regression is an important category of prediction problem where we have specific reasons to expect many current methods to perform less well [Foong et al., 2020]. Most importantly, it is possible to be extraordinarily wrong about regression in a way that is impossible with classification (assuming a constant misclassification utility). If we do stop using UCI datasets to evaluate Bayesian deep learning, we must be careful to also incorporate other regression problems into standard evaluations.

Dataset	MFVI	Radial	Dropout	VMG	FBNN	PBP_MV	DVI
Avg. Test LL and Std. Errors							
Boston	-2.58±0.06	-2.58±0.05	-2.46±0.25	-2.46±0.09	-2.30±0.04	02.54±0.08	-2.41±0.02
Concrete	-5.08±0.01	-5.08±0.01	-3.04±0.09	-3.01±0.03	-3.10±0.01	-3.04±0.03	-3.06±0.01
Energy	-1.05±0.01	-0.91±0.03	-1.99±0.09	-1.06±0.03	-0.68±0.02	-1.01±0.01	-1.01±0.06
Kin8nm	1.08±0.01	1.35±0.00	0.95±0.03	1.10±0.01	-	1.28±0.01	1.13±0.00
Naval	-1.57±0.01	-1.58±0.01	3.80±0.05	2.46±0.00	7.13±0.02	4.85±0.06	6.29±0.04
Pow. Plant	-7.54±0.00	-7.54±0.00	-2.80±0.05	-2.82±0.01	-	-2.78±0.01	-2.80±0.00
Protein	-3.67±0.00	-3.66±0.00	-2.89±0.01	-2.84±0.00	-2.89±0.00	-2.77±0.01	-2.85±0.01
Wine	-3.15±0.01	-3.15±0.01	-0.93±0.06	-0.95±0.01	-1.04±0.01	-0.97±0.01	-0.90±0.01
Yacht	-4.20±0.05	-4.20±0.05	-1.55±0.12	-1.30±0.02	-1.03±0.03	-1.64±0.02	-0.47±0.03
Avg. Test RMSE and Std. Errors							
Boston	3.42±0.23	3.36±0.23	2.97±0.85	2.70±0.13	2.38±0.10	3.11±0.15	-
Concrete	5.71±0.15	5.62±0.14	5.23±0.53	4.89±0.12	4.94±0.18	5.08±0.14	-
Energy	0.81±0.08	0.66±0.03	1.66±0.19	0.54±0.02	0.41±0.20	0.45±0.01	-
Kin8nm	0.37±0.00	0.16±0.00	0.10±0.00	0.08±0.00	-	0.07±0.00	-
Naval	0.01±0.00	0.01±0.00	0.01±0.00	0.00±0.00	0.00±0.00	0.00±0.00	-
Pow. Plant	4.02±0.04	4.04±0.04	4.02±0.18	4.04±0.04	-	3.91±0.14	-
Protein	4.40±0.02	4.34±0.03	4.36±0.04	4.13±0.02	4.33±0.03	3.94±0.02	-
Wine	0.65±0.01	0.64±0.01	0.62±0.04	0.63±0.01	0.67±0.01	0.64±0.01	-
Yacht	1.75±0.42	1.86±0.37	1.11±0.38	0.71±0.05	0.61±0.07	0.81±0.06	-

Table 4.2: Avg. test RMSE, predictive log-likelihood and s.e. for UCI regression datasets. Bold is where one model is better than the next best \pm their standard error. Results are from multiple papers and hyperparameter search is not necessarily consistent. MFVI and Radial are our implementations of standard MFVI and our proposed model respectively. Dropout results are from Gal [2016]. Variational Matrix Gaussian (VMG) results are from Louizos and Welling [2016]. Functional Bayesian Neural Networks (FBNN) results are from Sun et al. [2019]. Probabilistic Backpropagation Matrix Variate Gaussian (PBP_MV) results are from Sun et al. [2017]. Deterministic VI (DVI) results are from Wu et al. [2019].

4.4 Discussion and analysis

In section 4.2 we observed that the multivariate Gaussian distribution typically used in MFVI features a ‘soap-bubble’—almost all of the probability mass is clustered at a radius proportional to $\sigma\sqrt{D}$ from the mean in the large D limit (illustrated in Figure 4.1). This has two consequences in larger models. First, unless the weight variances are very small, a typical sample from the posterior has a high L_2 distance from the means. Second, because the mass is distributed uniformly over the hypersphere that the ‘soap-bubble’ clusters around, each sample from the multivariate Gaussian has a high expected L_2 distance from every other sample (similarly proportional to $\sigma\sqrt{D}$). This means that as σ and D grow, samples from the posterior are very different from each other, which we might expect to result in high gradient variance.

In contrast, in Radial BNNs the expected distance between samples from the posterior is independent of D for the dimensionality typical of neural networks. The expected L_2 distance between samples from a unit hypersphere rapidly tends to $\sqrt{2}$ as the number of dimensions increases. Since the radial dimension is also independent of D , the expected L_2 distance between samples from the Radial BNN is independent of D . This means that, even in large networks, samples from the Radial BNN will tend to be more representative of each other.

As a result, we might expect that the gradient variance is less of a problem. We consider this experimentally by examining several hypotheses:

MFVI variance The variance of gradient estimates for MFVI explodes as the standard deviation of the approximate posterior grows.

Radial variance Meanwhile, the variance of gradient estimates for Radial BNNs does not explode as the standard deviation of the approximate posterior grows.

MFVI problem The exploding variance of gradient estimates is harmful for MFVI training.

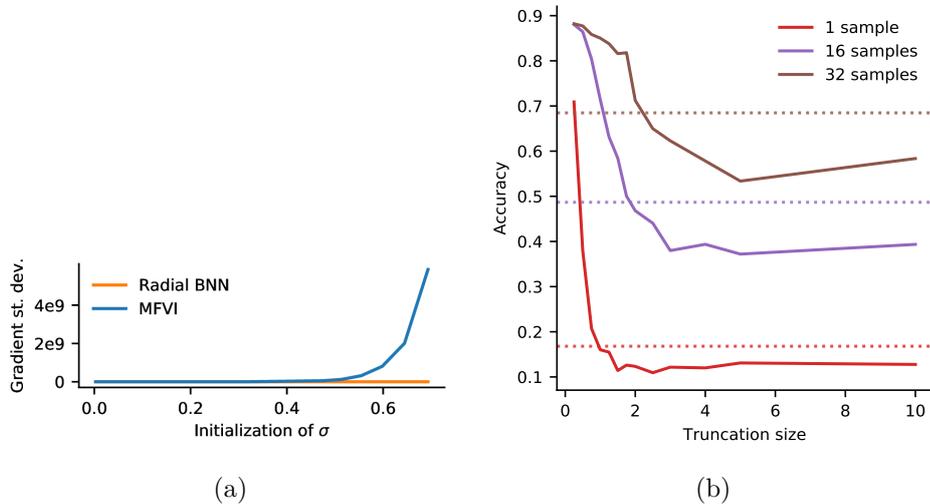


Figure 4.6: (a) The variance of gradient estimates in the standard MFVI posterior explodes as the weight variance parameter grows. (b) Dotted lines show untruncated Gaussian performance. Highly truncated Gaussians *improve* MFVI. This effect is most significant when small numbers of samples from the posterior are used to estimate the gradient. We conclude that despite bias, the low variance offered by truncation improves gradient estimates. Results averaged over 10 initial seeds for each truncation size.

MFVI mechanism The mechanism for the failure of MFVI is that the approximate posterior overfits to the loss reflecting the cross-entropy to the prior rather than the log-likelihood because it is low-variance even for large σ .

MFVI and Radial variance

We consider the first two hypotheses in fig. 4.6a. We show that for the standard MFVI posterior, the variance of initial gradients explodes after the weight standard deviation exceeds roughly 0.3. Meanwhile, for Radial BNNs this does not occur. This figure shows a 3x3 conv layer with 512 channels, but we observe a similar pattern regardless of the specifics of the architecture. This matters because, for MFVI with a unit Gaussian prior, the KL-divergence term of the loss is minimized by $\sigma_i = 1$ —well within the region where gradient noise has exploded.

MFVI problem

We can show that the gradient variance is hurting training for MFVI by using an alternative scheme that uses the same approximate posterior but has a low-variance and biased estimator of the gradient. We do this by truncating the sampling distribution using rejection sampling, selecting only samples from

a Gaussian distribution which fall under a threshold. Our new estimate of the loss is biased (because we are not sampling from the distribution used to compute the KL divergence) but has lower variance (because only samples near the mean are used).

In fig. 4.6b we show that the truncated models to outperform ‘correct’ MFVI with standard deviations initialized at $\sigma = 0.12$. Moreover, if our hypothesis is correct, smaller the number of variational samples should result in a higher variance and therefore a larger effect size. This is what we observe.

MFVI mechanism

Last, we examine the specific mechanism by which MFVI is failing due to the high variance gradient estimates. In Figure 4.7 we show a sample training run on the down-sampled version of the diabetic retinopathy dataset using an MFVI and Radial BNN with the same hyperparameters. Pathologically, the *training* accuracy falls for MFVI after about 150 epochs (top graph). The critical moment corresponds to the point where the training process begins to optimize the prior cross-entropy term of the loss, sacrificing the negative log-likelihood term (middle graph). We can further show that this corresponds to the point where the standard deviation of the negative log-likelihood term of the gradient begins to sharply increase for MFVI. Meanwhile, the prior cross-entropy term is computed analytically, so its variance does not grow as the values of σ increase during training from their tiny initializations.

This suggests that, MFVI fails in this setting because the high variance of the negative log-likelihood (NLL) term of the loss causes the optimizer to improve the cross-entropy term at the expense of the NLL term. For Radial BNNs, however, the NLL gradient variance stays low throughout training. Of course, it is possible that in other settings (architecture and data) the behaviour is different. The fact that Nado et al. [2021] do not find a benefit to using Radial BNNs in ResNets relative to their ‘tweaked’ MFVI might mean that the benefit in ResNets is smaller (although they do not compare to a vanilla implementation of MFVI).

In addition, it is possible that where an MFVI approximate posterior *is* able to optimize successfully, the requirement that samples from it behave somewhat similarly in function-space might result in especially flat minima, possibly resulting in better generalization.

Avoiding the Pathology in MFVI

Based on this analysis, we can see why Radial BNNs fix a sampling problem in MFVI. But this also helps explain why the ad-hoc tweaks which researchers have been using for MFVI have been successful. These tweaks chiefly serve to keep the weight variance low. Researchers initialize with small variances [Blundell et al., 2015, Fortunato et al., 2017, Nguyen et al., 2018]. Some-

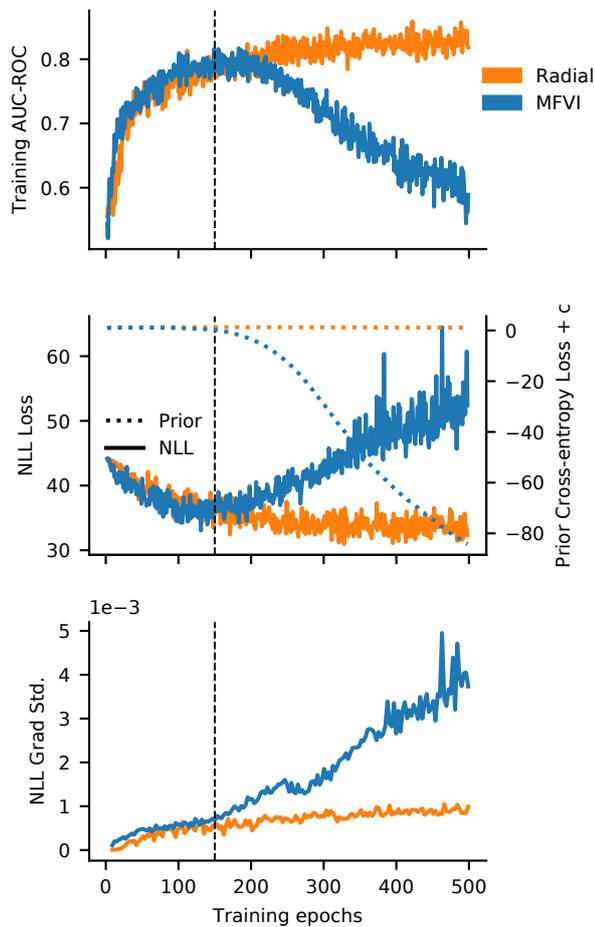


Figure 4.7: We can track the deterioration of the MFVI training dynamics. **Top:** After ~ 150 epochs (dashed line) *training* set performance degrades for MFVI while Radial continues to improve. **Middle:** for MFVI, the NLL term of the loss *increases* during training, but the prior cross-entropy term falls faster so the overall loss continues to fall. **Bottom:** The standard deviation of the NLL gradient estimator grows sharply for MFVI after about 150 epochs. This coincides with the moment where the loss is optimized by minimizing the prior cross-entropy and sacrificing the NLL.

times they adapt the loss function to remove or reduce the weight of the KL-divergence term, which reduces the pressure on weight variances to grow [Fortunato et al., 2018]. Other times researchers pre-train the means with just the NLL loss, which makes it possible to stop training after relatively little training on the ELBO loss, which stops the variances from growing too much [Nguyen et al., 2018]. Another approach, which we have not seen tried, would be to use a very tight zero-centred prior, effectively enforcing the desire to have a basically deterministic network (a prior inversely proportional to \sqrt{D} would balance the ‘soap-bubble’ variance). However, this sort of very tight prior is not compatible with the use of data-dependent priors in sequential learning. Nado et al. [2021] instead centre the ‘prior’ at the current mean of the approximate posterior and move the ‘prior’ during training, which they find performs well although it departs from the standard motivation for priors.

For most of these tweaks, the resulting network is not fully optimizing the ELBO. This does not necessarily make the resulting network useless—after all, the ELBO is only a bound on the actual model evidence. However, if we have a theoretically principled way to fix our sampling problems without resorting to ad-hoc tweaks, then we should prefer that. Radial BNNs offer exactly that theoretically principled fix.

4.5 Conclusion

In this chapter, we have demonstrated how the choice of approximating distribution can interact with the optimization procedure for approximate Bayesian methods that employ optimization. We demonstrated an interaction in mean-field variational inference caused by the interaction between optimization and approximation, which are independent of the hypothesized restrictiveness of the mean-field approximation discussed in the previous chapter. By examining this interaction, we have proposed an alternative approximating distribution whose optimization and prediction are computationally equivalent to the multivariate Gaussian approximate posterior. In different architectures and datasets, it is possible that we would not observe the same interactions that we observed here. In that case, the proposed fix would be less likely to work—but it would nevertheless demonstrate the same core point: that architecture, data, and approximation scheme can interact in surprising ways that affect the quality of the Bayesian approximation.

Chapter 5

Evaluating Bayesian Deep Learning

Evaluating Bayesian deep learning is difficult. In some sense, the ideal evaluation could be a comparison between an approximate and true posterior distribution but, as we discuss in chapter 2, establishing a satisfactory distance measure is challenging (c.f. 2.3). Even setting this challenge aside, HMC remains our best tool for sampling from the true posterior and it is difficult and expensive for modern architectures although high-quality reference samples exist for, for example, ResNets and have been used for benchmarking evaluations [Wilson et al., 2021].

In the context of model misspecification, Key et al. [1999] propose casting model evaluation as a Bayesian decision problem and evaluating the expected utility of the posterior distribution on a ‘generic’ utility function, such as the log-likelihood. In fact, this can be seen as a solution to the approximate inference conundrum as well. The trouble is that different applications interact with nuances of the approximations, which makes a generic utility less appropriate.

In this chapter, we consider two archetypal use cases for Bayesian deep learning which test different aspects of the ‘Bayesian’ procedure. Both of these build off what I see as the key characteristic of approximate Bayesian methods: that the parameter distributions of our models reflect our beliefs about states of affairs. First, we consider continual learning as a test of the ability of an approximate posterior to be used as part of a sequential Bayesian updating scheme. Second, we consider active learning as a test of the ability of an approximate posterior to represent its subjective epistemic state. Both of these are key properties associated with the Bayesian approach.

A natural way to evaluate approximate Bayesian inference, therefore, is to evaluate the use a Bayesian method to accomplish these two tasks. For continual learning, this might mean adopting a specific Bayesian approximation, using it for sequential inference, and evaluating the methods using the performance of the final model over a sequence of tasks. For active learning, it might mean estimating the expected information gain under a Bayesian approximation of the posterior to acquire data and measuring the performance of the final model.

Unfortunately, neither of these should be regarded as an effective evaluation of approximate inference. Specifically, we show that most existing evaluations of continual learning ‘abuse’ the evaluation structure to *bypass* the Bayesian approximation. We identify one effective evaluation setting, but note that existing posterior approximations perform extremely badly on it. We further show that active learning performance depends significantly on implicit biases introduced by the acquisition procedure, and that the extent of this bias is different for different models in a manner which does not depend on the quality of the approximate posterior. This recalls the results of chapter 3 and chapter 4 which showed how the action of the approximation scheme depends on the model architecture; here even the action of the *evaluation* scheme depends on the architecture. Removing the bias at least removes one source of differing performance between models that is unrelated to Bayesian inference. However, we do not guarantee that active learning with the bias removed might not suffer from some *other* confounding factor when used as an evaluation of approximate Bayesian inference.

5.1 Continual Learning

Bayesians do not all agree on what exactly ‘Bayesianism’ entails. One consistent element, which is especially clear in the work of Jaynes [2003] and Cox [1961], is the requirement that a Bayesian’s subjective beliefs ought to follow the laws of probability as further observations are obtained. At the same time, an explicit axiom in some cases, a Bayesian’s subjective beliefs ought to depend only on their prior and the observations and not on the order in which the data were processed.

All of these things are put under strain in continual learning, a challenge which poses significant difficulties for Bayesian inference in neural networks.

The Continual Learning Challenge

Continual learning requires that a model is trained on multiple datasets arriving in series, each of which is then forgotten before the next dataset arrives [Kirkpatrick et al., 2017, Zenke et al., 2017]. These datasets are often called *tasks*—although this term is loaded in a way that is not enforced in the formalism. The goal is that the model learns to perform as well as it would have

had it trained on the entire dataset at the same time, which involves both avoiding forgetting information learned on earlier as well as transferring information from early tasks forwards [Chaudhry et al., 2018]. The test dataset is therefore generally, but not always, treated as a balanced mixture of the individual task distributions.

Continual learning is related to a number of other interesting machine learning problems.

Online learning [Opper, 1999] differs in that data arrive in sequence one-by-one. This is equivalent to continual learning in the special case that each task is a single datapoint.

Meta learning [Rendell et al., 1987] differs in that the multiple datasets are available at the same time and that the test distribution is generally separate from the training tasks in some way.

Multi-task learning [Caruana, 1997] differs in that the multiple datasets are available at the same time.

Curriculum learning [Bengio et al., 2009] differs in that the goal of curriculum learning is to actively construct the tasks/datasets so that the performance is *better* than it would have been had the model been trained on all the data together. (Note that curriculum learning therefore generally *relies* on the failure of neural network training to be indifferent to data-ordering.)

Continual learning can therefore be thought of as a form of softly online multi-task learning.

Although it is widely contended that continual learning is of practical value, in fact this is not very clear. Foundational work on the subject [Zenke et al., 2017, Kirkpatrick et al., 2017] was motivated more by curiosity about the phenomenon of catastrophic forgetting [Robins, 1995, Choy et al., 2006, Goodfellow et al., 2013] which occurs when neural networks are trained on first one dataset and then another. (In fact, ‘catastrophic forgetting’ is in some sense the obvious consequence of optimizing the parameters with respect to a different objective and the shocking and interesting phenomenon ought to be considered to be *magical remembering*, which is a consequence of imperfect local optimization.)

Continual learning has been proposed as valuable for adapting to changing environments over time [Lomonaco et al., 2020], or as useful for robotics and industrial control. In fact these settings are often more cleanly online multi-task learning because real robotic applications rarely provide data in clearly demarcated sequential sets. The difficulty of being precise about what exactly is needed in the ‘continual’ learning setting causes difficulties for evaluating progress in the field. One case where regulatory constraints might demand

continual learning is when privacy laws require data to be deleted but allow the storage of models derived from the data, which has motivated differentially private continual learning [Farquhar and Gal, 2018a].

Partly because of the lack of clear agreement about the intended use case for continual learning, there is considerable variation in the evaluations which are employed and the constraints which are enforced. Our goal in this chapter is not to arbitrate which constraints and assumptions are most useful for some application or another. Rather, our goal is to identify a set of constraints that provides the most interesting measure of the quality of an approximate posterior.

Formally, in a typical supervised learning setting, we aim to learn parameters \mathbf{w} using an independently and identically distributed (i.i.d.) labelled training dataset $\mathcal{D} \equiv \{(\mathbf{x}_i, y_i)\}$ to accurately predict $p(\hat{y}_i | \mathbf{x}_i; \mathbf{w})$. Instead, for continual learning we split the data into T disjoint subsets $\mathcal{D}^t \equiv \{(\mathbf{x}_i^t, y_i^t)\}$ which are individually i.i.d. and represent a task.

Researchers often make several further assumptions, but not consistently. Some of these are discussed in more detail in later sections, but amongst the most common decisions are:

- Is the model architecture is fixed? Or are additional parameters available for each new task?
- Is the task identity is known during training, during testing, or for both?
- Can a small amount of data (a coreset) be carried forwards? Carried backwards? Or must everything be lost?
- Can further data products (e.g., gradients) be carried between tasks?

The answers to these questions are absolutely pivotal, and many continual learning methods rely entirely on the ability to exploit one or more of these ‘loopholes’ in the set-up. Further assumptions are often made about the relationship between the tasks, and these will be considered in more detail below.

Continual Learning as Bayesian Sequential Learning

By considering the weights of our neural network as subjective belief distributions, we can frame continual learning as a Bayesian updating problem, following [Nguyen et al., 2018]. The posterior given two tasks is, using Bayes’ theorem

$$p(\mathbf{w} | \mathcal{D}^1, \mathcal{D}^2) = \frac{p(\mathcal{D}^1, \mathcal{D}^2 | \mathbf{w})p(\mathbf{w})}{p(\mathcal{D}^1, \mathcal{D}^2)}, \quad (5.1)$$

and assuming that the task data are conditionally independent of each other,

$$= \frac{p(\mathcal{D}^2 | \mathbf{w})p(\mathcal{D}^1 | \mathbf{w})p(\mathbf{w})}{p(\mathcal{D}^1, \mathcal{D}^2)}, \quad (5.2)$$

and lastly applying Bayes’ theorem a second time

$$= \frac{p(\mathcal{D}^2 | \mathbf{w})p(\mathbf{w} | \mathcal{D}^1)}{p(\mathcal{D}^2 | \mathcal{D}^1)}. \quad (5.3)$$

This is identical to Bayes’ theorem for $p(\mathbf{w} | \mathcal{D}^2)$ except that the ‘prior’ distribution for \mathbf{w} is given by the posterior on the first dataset, $p(\mathbf{w} | \mathcal{D}^1)$, and that the normalization term is now conditioned on the prior datasets. That is to say, the posterior given both tasks is the same as the posterior given the second task where we already use the first task’s posterior as the prior. Nguyen et al. [2018] propose using approximate inference for this latest Bayesian updating step, and further observed that the decomposition can be applied recursively for arbitrarily many tasks. They call their method Variational Continual Learning (VCL). This generalizes the method of Kirkpatrick et al. [2017] which, when combined with the comment by Huszar [2018], can be interpreted as a Laplace approximation for Bayesian sequential learning. Nguyen et al. [2018] further use coresets to improve their method’s performance, which turn out to be essential, but when we discuss “VCL” we mean to refer to the version of the method without coresets, and we will be explicit when coresets are additionally used.

The relationship is exact. If we had access to the exact posterior, it would be precisely the same to compute the posterior step-wise rather than all at one time. Indeed, it would be identical regardless of the order in which we performed the inference.

Note that we can choose to adopt a Bayesian perspective about both the parameter distribution and the data distribution [Farquhar and Gal, 2018b]. In what follows, we will assume that we are trying to evaluate the quality of a predictive parametric distribution, which makes a framework like VCL appropriate. If the goal were to evaluate the quality of a generative model, an alternative continual learning algorithm would be more appropriate.

Inadequate Evaluations of Continual Learning

Several evaluations are widely used in the literature despite being extremely limited as evaluations for continual learning algorithms, and which are also poor evaluations for Bayesian deep learning as a solution to continual learning. In this section, we discuss two of the most common flawed evaluations: permuted MNIST, and multi-headed split evaluations. These are flawed evaluations of continual learning generally, and when continual learning is set up as an evaluation of Bayesian sequential updating they become flawed evaluations for approximate Bayesian inference. In recent years, research practice has often recognized the shortcomings identified in this section, which were novel at the time of publication.

Permuted MNIST

The Permuted MNIST experiment was introduced by Goodfellow et al. [2013]. In their experiment, a model is trained on MNIST as \mathcal{D}_1 . Each later \mathcal{D}^t for $1 < t \leq 10$ is constructed from the MNIST data but with the pixels of each digit randomly permuted. A fresh permutation is drawn for each task and applied to all images in the same way for that task. After training on each dataset, one evaluates the model on each of the previous datasets as well as the current. Goodfellow et al. [2013] used this experiment to investigate feature extraction, but it has since become widely used for continual learning evaluation [Zenke et al., 2017, Kirkpatrick et al., 2017, Lee et al., 2017, Lopez-paz and Ranzato, 2017, Nguyen et al., 2018, Ritter et al., 2018, Hu et al., 2019, Chaudhry et al., 2018].

Unfortunately, permuted MNIST represents an unrealistic best case scenario for continual learning—although it satisfies the literal definition of continual learning. The positions of the pixels are fully randomized—which suited its original purpose. Goodfellow et al. [2013] investigated whether neural networks have ‘high-level concepts’ that get re-mapped to pixel positions when the input space is permuted, and found evidence they did not. Now, however, the experiment has been repurposed for continual learning, where it is not suitable.

An image from each permuted dataset is practically unrecognizable given previous datasets. The actual world is almost never structured like this—new situations look confusingly similar to old ones. A model presented with a new task in Permuted MNIST will be uncertain, while in settings that are not deliberately randomized a model will tend to make confident but false predictions. This makes Permuted MNIST significantly different from real-world settings in a way that directly affects how new tasks are learned. As an illustration of this, we find that the average entropy of predictions for the second task of permuted MNIST is 0.45 while for a slightly more realistic task (split MNIST, discussed in section 5.1) the entropy is 0.003.

Moreover, from the perspective of Bayesian inference, permuted MNIST represents an atypical inference task. Because each task is so distinct, the likelihood term of the ELBO loss is similar in magnitude for all datapoints, resulting in a strong contribution from the prior term. In contrast, for more natural data distributions, the likelihood term is very high in early training steps because of confident but incorrect predictions. This reduces the regularization of the prior early in training, leading the approximate posterior to leave the local optimum for the previous task and resulting in a less suitable approximation to the joint posterior.

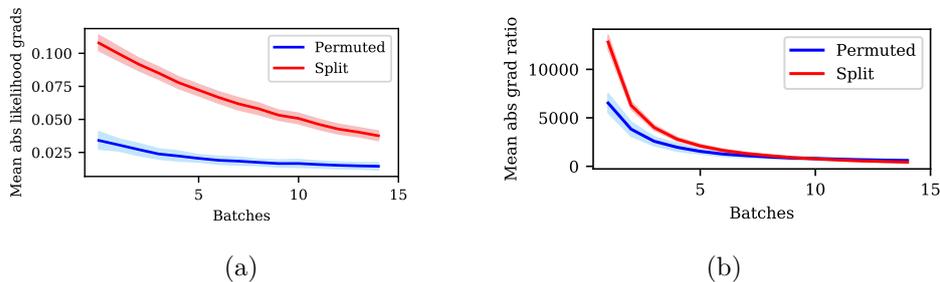


Figure 5.1: Early in the second task, because permuted MNIST tasks are so disjoint, (a) the likelihood term of gradients on the final layer is unusually low (b) relative to the prior term. This makes continual learning unrealistically easy in this evaluation. Averaged over 100 runs, shading is one standard deviation.

‘Multi-headed’ Continual Learning

The Split MNIST experiment was introduced by Zenke et al. [2017] in a *multi-headed* form and used by other authors including Shin et al. [2017], Nguyen et al. [2018], Chaudhry et al. [2018] (Ritter et al. [2018] use a two-task variant). A number of variants have later emerged which use other datasets such as FashionMNIST, CIFAR, and Core50. However, although they increase the difficulty of function approximation, these variants do not significantly affect the dynamics we describe here. The experiment constructs a series of five related tasks. The first task is to distinguish the digits (0, 1), then (2, 3) etc.

Most papers use a multi-headed variant the model prediction is constrained to be only from the two classes represented in each task. For example, when evaluating the performance on the first task, the model only needs to predict probabilities for zero versus one. In some cases, multi-heading is taken even further and training is only done on the head governing the specific classes included in the task [Zenke et al., 2017, Nguyen et al., 2018, Ritter et al., 2018]. This effectively means that each task has a dedicated task-specific component of the parametric model.

A *single-headed* version does not limit predictions during either training or testing. As Chaudhry et al. [2018] note, the multi-headed variant is much easier to solve, but it requires knowledge of the task and which classes are represented in each task. Multi-heading is often used in similar non-MNIST evaluations, for example, in Zenke et al. [2017], Nguyen et al. [2018], Ritter et al. [2018]. Chaudhry et al. [2018] use both a single- and multi-headed version of Split MNIST. Hu et al. [2019] use a single-headed set-up, but their method effectively learns a different head for each task.

Unfortunately, Bayesian methods that focus on parameter distributions of predictive functions for continual learning tend to look much better in multi-headed evaluations than single-headed ones. Suppose some model has already been trained on the first four tasks of Split MNIST and is then tested on the digit ‘1’. In the single-headed variant, when shown a ‘1’ it may incorrectly

predict the label is seven, which was seen more recently. In the multi-headed variant, we *knowingly assume* that the label comes from [0:1]. Because the model only needs to decide between 0 and 1 (and not even consider if the image is a 7), a multi-headed model could correctly predict the label is 1 even though the same approach will completely fail in a single-headed experiment setup. That is to say, solving the multi-headed split MNIST problem does not require actually learning the correct posterior over the weights conditioned on the joint dataset, $p(\mathbf{w} \mid \mathcal{D}^0, \dots, \mathcal{D}^t)$, or even the slightly weaker requirement of learning the predictive posterior, $p(y_i^t \mid \mathbf{x}_i^t, \mathcal{D}^0, \dots, \mathcal{D}^t)$. Instead, it requires the much weaker condition of learning a predictive distribution with an additional task label $p(y_i^t \mid \mathbf{x}_i^t, t, \mathcal{D}^0, \dots, \mathcal{D}^t)$.

The problem is, in fact, even deeper. If we again consider eq. (5.3), in which we derived the Bayesian recurrence relation which allows Bayesian continual learning, a key step was to assume that the quantity \mathbf{w} was *the same* in our two probabilistic expressions! However, in the multi-headed setting, we are in fact learning *multiple functions* with overlapping parameterization. Multi-headed VCL learns f^1 parameterized by a shared \mathbf{w} as well as task-specific \mathbf{w}^1 and then learns f^2 with \mathbf{w} and \mathbf{w}^2 . So beginning as we did before we have

$$p(\mathbf{w}, \mathbf{w}^1, \mathbf{w}^2 \mid \mathcal{D}^1, \mathcal{D}^2) = \frac{p(\mathcal{D}^1, \mathcal{D}^2 \mid \mathbf{w}, \mathbf{w}^1, \mathbf{w}^2)p(\mathbf{w}, \mathbf{w}^1, \mathbf{w}^2)}{p(\mathcal{D}^1, \mathcal{D}^2)}, \quad (5.4)$$

and with the same conditional independence assumption,

$$= \frac{p(\mathcal{D}^2 \mid \mathbf{w}, \mathbf{w}^1, \mathbf{w}^2)p(\mathcal{D}^1 \mid \mathbf{w}, \mathbf{w}^1, \mathbf{w}^2)p(\mathbf{w}, \mathbf{w}^1, \mathbf{w}^2)}{p(\mathcal{D}^1, \mathcal{D}^2)}. \quad (5.5)$$

And now we would like to show, using the ‘reverse’ Bayes rule argument from before, that this is proportional to

$$\propto p(\mathcal{D}^2 \mid \mathbf{w}, \mathbf{w}^2)p(\mathbf{w}, \mathbf{w}^1 \mid \mathcal{D}^1). \quad (5.6)$$

This would follow if \mathcal{D}^i and \mathcal{D}^j were conditionally independent given \mathbf{w}^j for $i \neq j$ and vice versa. But that is not true, and should not in general be expected to be approximately true for datasets where we expect tasks to be related to each other. That is to say, if multi-headed continual learning is to be used as an evaluation for Bayesian approximate inference, it must be with respect to a much more sophisticated probabilistic model than any current methods use.

Nevertheless, in practice, good multi-headed performance is significantly easier to achieve than single-headed performance, as we show below, which has led many papers to continue to evaluate themselves in this way. A multi-headed evaluation can therefore make it seem as if an approach has solved a continual learning problem when it has not.

Good Evaluations of Bayesian Deep Learning with Continual Learning

A good evaluation of Bayesian inference with continual learning ought to test the Bayesian performance of the part of the scheme which does the ‘heavy lifting’. As an extreme example, a ‘coreset’ method which stored the entirety of all past datasets in order to perform variational inference would not demonstrate the effectiveness of the Bayesian approximation.

The single-headed split MNIST evaluation is a passable test of the performance of an approximate Bayesian posterior when the method employed does not use a coreset. We show below, however, that using a coreset allows almost perfect performance *even without anything else at all* and that past evaluations of methods like VCL have inadvertently disguised their true effectiveness by relying on coresets.

We use three variants of VCL. First, we use pure VCL without a coreset, exactly as described by Nguyen et al. [2018]. This represents a pure prior-focused approach. Second, we use a small coreset of 40 datapoints extracted from each dataset (we use their k-center coreset approach, rather than a randomly selected one, but as they note this does not have a large effect). The second method is the same as pure VCL except that, at the end of training on each task, the model is trained on the coresets before testing, as described in their work. This reflects a hybrid approach. Third, we use a ‘coreset only’ approach. It is exactly like the second variant except that the prior used for variational inference is the initial prior each time—it is not updated after each task.¹

We use two further baselines to show that the effects we find are not an artifact of VCL specifically. EWC [Kirkpatrick et al., 2017] is used as a Bayes-inspired alternative to VCL which can be interpreted as Bayesian continual learning using a Laplace-approximation rather than variational inference. Last, we compare to a more powerful ‘data-space’ approach called VGR [Farquhar and Gal, 2018b] which in fact side-steps the problem of learning the posterior distribution over the parameters of the neural network by relying on generative models of the data to estimate the data-likelihood term of the ELBO loss. VGR, VCL and its variants use a Bayesian neural network (BNN) and variational inference, while EWC does not, so performance comparisons between these architectures should be interpreted carefully.

The performance of VCL with coreset appears to be entirely driven by the presence of the coresets (see fig. 5.2a and fig. 5.2b). When coresets are removed, VCL alone completely forgets old tasks—its accuracy comes from correctly classifying the most recent task only. (By chance, a model with good accuracy on only the latest task will get roughly 100%, then 50%, 33% etc. which is exactly what we see.) EWC performs exactly like VCL. This suggests

¹This is not the coreset only algorithm used in Nguyen et al. [2018]. Theirs is seeing *only* coresets—much less data—which is why it performs badly on even the first task.

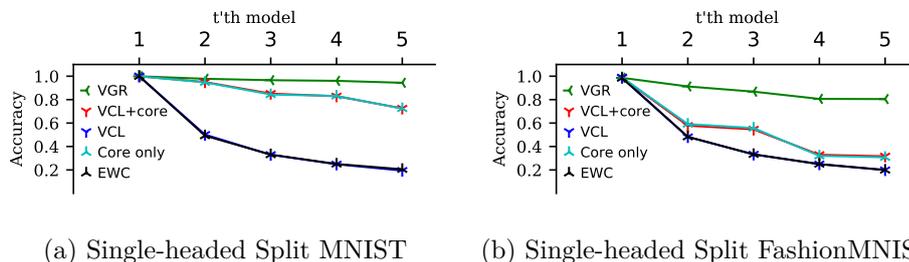


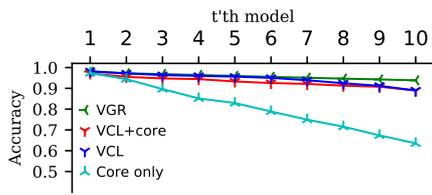
Figure 5.2: Single-headed Split set-ups reveal the failure of Bayesian sequential updating to prevent catastrophic forgetting.

that neither mean-field variational inference with a multi-variate Gaussian posterior nor the Laplace approximation are succeeding in capturing enough of the posterior to allow sequential Bayesian updating. Moreover, it suggests that neither is getting anywhere close.

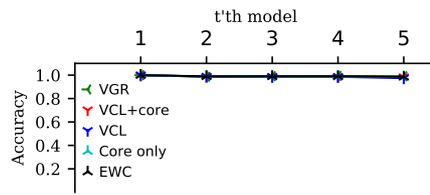
Using FashionMNIST rather than MNIST is harder and a worthwhile additional test, but does not reveal a radically different story. VCL performs much worse on FashionMNIST than VGR even though both use the same model.

Contrast to Inadequate Evaluations

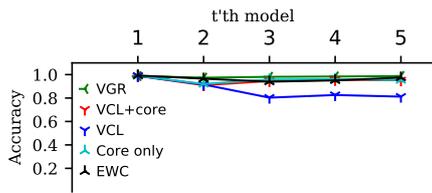
To demonstrate the failure of inadequate evaluations to identify shortcomings of the approximate inference procedure, we show the performance of these methods on permuted MNIST/FashionMNIST and multi-headed split MNIST/FashionMNIST which assumes access to labels during either training and testing or just testing. In fig. 5.3 we show that none of these evaluations succeeds in distinguishing the over-reliance of ‘Bayesian’ methods on coresets. The ‘multi-headed’ versions are trained and tested using only the ‘active’ head, assuming task knowledge. In contrast, the ‘test-time knowledge’ versions are trained in the same way as a single-headed network but assume knowledge of the task label at test-time.



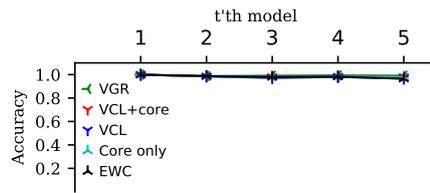
(a) Permutated MNIST



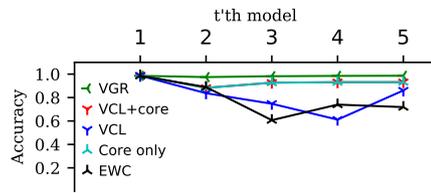
(b) Multi-headed MNIST



(c) Multi-headed FashionMNIST



(d) Test-time Knowledge MNIST



(e) Test-time Knowledge FashionMNIST

Figure 5.3: The inadequate evaluations discussed in section 5.1 do not succeed in distinguishing the over-reliance of ‘Bayesian’ methods on coresets.

5.2 Active Learning

Statement of contribution: *the work in this section draws heavily on [Farguhar et al., 2021] which was written with equal contribution by Tom Rainforth. In particular, Tom was chiefly responsible for the proofs of the unbiasedness and consistency of \tilde{R}_{PURE} and \tilde{R}_{LURE} and the analysis of their variance. For this reason, rather than restating those results in this thesis, I just cite the original paper for their properties.*

Active learning aims to improve learning label-efficiency by selecting informative points to label [Atlas et al., 1990, Settles, 2010]. Although not all approaches to active learning are grounded in information theory (e.g., [Sener and Savarese, 2018]), a number of promising approaches have taken a Bayesian approach and interpreted the problem of active learning as estimating the expected information gain about the subjective belief distributions over the network’s parameters when new labels are acquired [MacKay, 1992b, Houlby et al., 2011]. This suggests that the effectiveness of an approximate posterior at supporting active learning might be a good test of its approximation of the true posterior. These methods have been applied to deep learning, especially in computer vision [Gal et al., 2017b, Wang et al., 2017]

A naive approach to evaluating approximate Bayesian inference methods would work as follows:

1. Define some approximate Bayesian inference scheme, a dataset, and a probabilistic model.
2. Perform approximate inference given an initial dataset.
3. Select new points to acquire in order to expand the dataset using Bayesian Active Learning by Disagreement (BALD) [Houlby et al., 2011], which estimates the expected information gain about parameter distributions from acquiring newly labelled points [MacKay, 1992b].
4. Repeat the procedure some number of times, and then repeat the whole thing for several approximate inference schemes.
5. Compare the performance of the resulting models. We then evaluate the performance of our approximate inference scheme based on the method that produced the best problem-specific outcome.

Unfortunately, this will not work as a reliable evaluation of approximate inference in Bayesian neural networks. In particular, we show that the performance of active learning relies partly on the presence of an implicit bias which is introduced by the active sampling scheme which has nothing to do with the posterior approximation. A ‘bad’ posterior approximation can guide the model towards useful data-points by representing implicit ‘prior knowledge’

which is not encoded directly in the Bayesian problem, but gradually emerges from natural-selection of research.

We demonstrate this by showing how to remove the bias, and show that this is harmful in some cases while it is helpful in others. We argue that the reasons for this are related to the presence of over-fitting and interpolation, rather than the quality of the Bayesian inference approximations. In fact, we show that removing the bias can change the ordering of which approximation scheme appears superior, without having any effect on the underlying posterior approximation.

Removing this confounding factor may allow *active testing* [Kossen et al., 2021], rather than active learning, to be a more effective evaluation of the quality of approximate inference. However, this conclusion can only be tentative, because there may be other shortcomings of active testing as an evaluation scheme which have not yet been identified.

Bias in Active Learning

Active learning introduces a bias to the Monte Carlo estimation of the risk or of the loss used during training a neural network or Bayesian neural network. Intuitively, this comes from the fact that the data distribution is no longer identically and independently distributed (i.i.d.) following the population distribution, and that the data distribution is not independent of the parameter distribution. This sampling bias has been noted and discussed by, for example, MacKay [1992b] who dismissed it from a Bayesian perspective based on the ‘likelihood principle’—that, given a statistical model, the likelihood is a complete summary of the evidence, regardless of how the data was acquired. However, the principle remains controversial [Rainforth, 2017] because there do seem to be situations where the manner in which the data are gathered ought to influence inferences and in this case applying the likelihood principle would assume a well-specified model. As a result, the likelihood is arguably *not* a complete summary of the evidence. Sampling bias has also been considered by Dasgupta and Hsu [2008], Beygelzimer et al. [2009], Chu et al. [2011], Ganti and Gray [2012].

Let us characterize the bias introduced by active learning more formally. In supervised learning, generally, we aim to find a decision rule f_{θ} corresponding to inputs, \mathbf{x} , and outputs, y , drawn from a population data distribution $p_{\text{data}}(\mathbf{x}, y)$ which, given a loss function $\mathcal{L}(y, f_{\theta}(\mathbf{x}))$, minimizes the population risk:

$$r = \mathbb{E}_{\mathbf{x}, y \sim p_{\text{data}}} [\mathcal{L}(y, f_{\theta}(\mathbf{x}))]. \quad (5.7)$$

The population risk cannot be found exactly, so instead we consider the empirical distribution for some dataset of N points drawn from the population. This gives the empirical risk: an unbiased and consistent estimator of r when

the data are drawn i.i.d from p_{data} and are independent of θ ,

$$\hat{R} = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(y_n, f_{\theta}(\mathbf{x}_n)). \quad (5.8)$$

In pool-based active learning [Lewis and Gale, 1994, Settles, 2010], we begin with a large unlabeled dataset, known as the pool dataset $\mathcal{D}_{\text{pool}} \equiv \{\mathbf{x}_n | 1 \leq n \leq N\}$, and sequentially pick the most useful points for which to acquire labels. The lack of most labels means we cannot evaluate \hat{R} directly, so we use the *sub-sample empirical risk* evaluated using the M actively sampled labelled points:

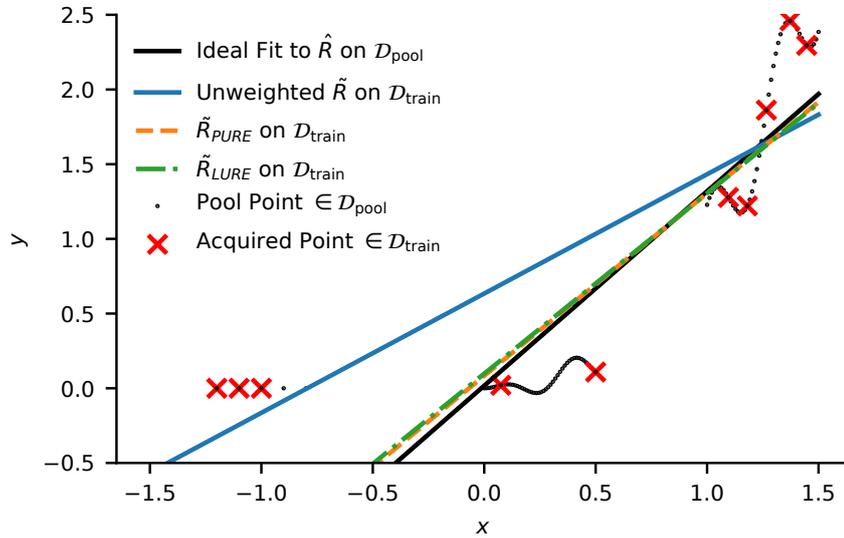
$$\tilde{R} = \frac{1}{M} \sum_{m=1}^M \mathcal{L}(y_m, f_{\theta}(\mathbf{x}_m)). \quad (5.9)$$

Though almost all active learning research uses this estimator (see Appendix C.2), it is not an unbiased estimator of either \hat{R} or r when the M points are actively sampled. Under active—i.e. non—uniform—sampling the M datapoints are not drawn from the population distribution, resulting in a bias.

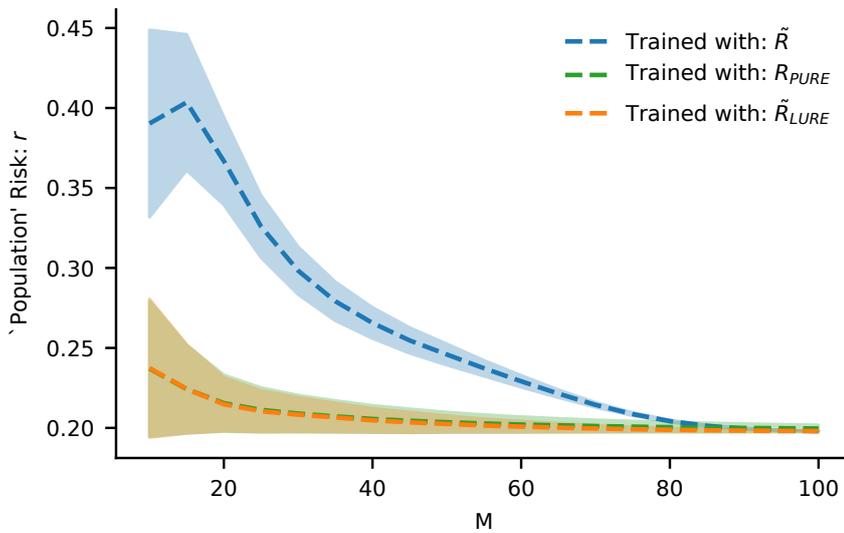
Note an important distinction between what we will call “statistical bias” and “overfitting bias.” The bias from active learning above is a statistical bias in the sense that using \tilde{R} biases our estimation of r , regardless of θ . As such, optimizing θ with respect to \tilde{R} induces bias into our optimization of θ . In turn, this breaks any consistency guarantees for our learning process: if we keep M/N fixed, take $M \rightarrow \infty$, and optimize for θ , we no longer get the optimal θ that minimizes r . Almost all work on active learning for neural networks currently ignores the issue of statistical bias.

However, even without this statistical bias, indeed even if we use \hat{R} directly, the training process itself also creates an overfitting bias: evaluating the risk using training data induces a dependency between the data and θ . This is why we usually evaluate the risk on held-out test data when doing model selection. Dealing with overfitting bias is beyond the scope of our work as this would equate to solving the problem of generalization. The small amount of prior work which does consider statistical bias in active learning entirely ignores this overfitting bias without commenting on it.

We mostly focus on statistical bias in active learning, so that we can produce estimators that are valid and consistent, and let us optimize the intended objective, not so they can miraculously close the train–test gap. From a more formal perspective, our results all assume that θ is chosen independently of the training data; an assumption that is almost always (implicitly) made in the literature. This ensures our estimators form valid objectives, but also has important implications that are typically overlooked. We return to this in §5.2, examining the interaction between statistical and overfitting bias.



(a) Illustration with toy data.



(b) Test MSE. Linear regression

Figure 5.4: **(a)** Active learning deliberately over-samples unusual points (red x's) which no longer match the population (black dots). Common practice uses the biased unweighted estimator \tilde{R} which puts too much emphasis on unusual points. Our unbiased estimators \tilde{R}_{PURE} and \tilde{R}_{LURE} fix this, learning a function using only \mathcal{D}_{train} nearly equal to the ideal you would get if you had labels for the whole of \mathcal{D}_{pool} , despite only using a few points. **(b)** Removing the bias greatly improves the error of linear models fit to the data, with near-optimal performance using very few examples.

Active Learning Is An Ineffective Posterior Evaluation

In this thesis, we will help ourselves to two unbiased risk estimators, \tilde{R}_{PURE} and \tilde{R}_{LURE} , as an alternative to \tilde{R} . These estimators are introduced by Farquhar et al. [2021], who prove that they are unbiased and consistent and that, under mild assumptions, they have lower variance than \tilde{R} . Using these estimators, we will show that the statistical bias introduced by active-learning sampling has an impact on performance evaluations for active learning despite not directly representing the quality of the approximate inference scheme.

Empirical Investigation of Removing Bias During Active Learning

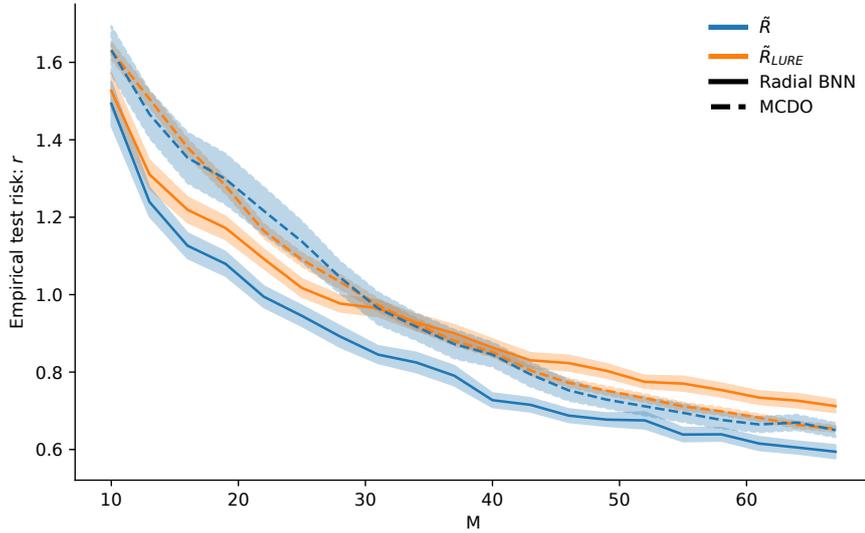
Intuitively, removing bias in training while also reducing the variance ought to improve the downstream task objective: test loss and accuracy. To investigate this, we train models using \tilde{R} , \tilde{R}_{LURE} , and \tilde{R}_{PURE} with actively sampled data and measure the population risk of each model.

On a toy linear regression task (Figure 5.4), our estimators improve the test loss—even with small numbers of acquired points we have nearly optimal test loss. Here, therefore, removing active learning bias greatly improves downstream performance despite having *no* effect on the inference, which is analytical.

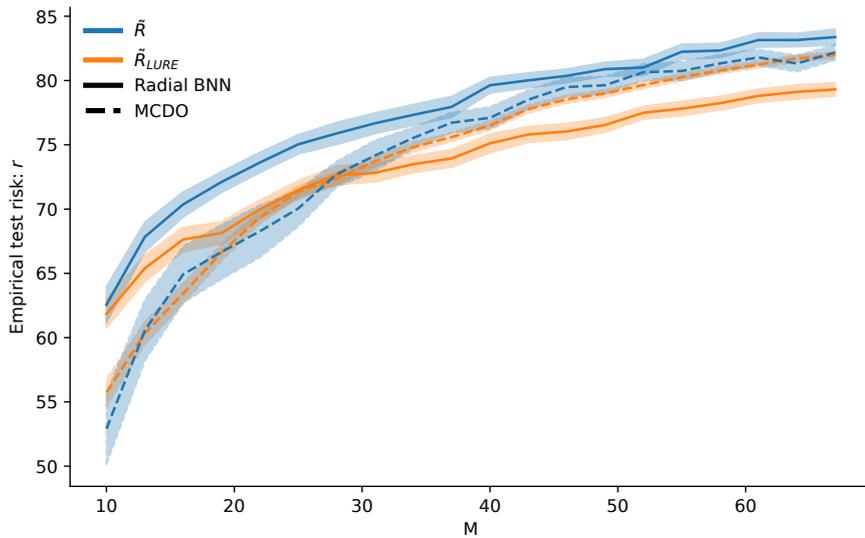
But we further investigate FashionMNIST and MNIST using a Bayesian neural network trained either with the Radial approximating distribution or Monte Carlo dropout (MCDO) [Gal and Ghahramani, 2015]. This demonstrates that removing the bias does not always improve active learning performance. We use an unbalanced variant of both datasets which should allow active learning to shine, though note that we check results on the standard set as well in the appendix. Following Kirsch et al. [2019], we use a ‘consistent’ version of MCDO which reduces score variance.

In all cases there is a small but significant negative impact on the full test dataset loss of training with \tilde{R}_{LURE} or \tilde{R}_{PURE} and a slightly larger negative impact on test accuracy (fig. C.3). That is, we get a better model by training using a biased estimator with higher variance! The reasons for this are discussed in section 5.2.

This effect is strong for Radial BNNs and comparatively weak for MCDO, especially as more data is acquired. For MNIST, the biased version of Radial BNNs performs slightly better on both accuracy and negative-log-likelihood than the biased version of MCDO, while the unbiased version of MCDO performs noticeably better than the unbiased version of the Radial BNN. That is to say, removing the bias in label-acquisition changes the ordering of which of these two approximate inference schemes appears better, despite having no direct effect on the quality of the posterior approximation. This demonstrates the failure of active learning to serve as an effective evaluation of posterior approximation.



(a) Test loss.



(b) Test accuracy.

Figure 5.5: **MNIST**. Training with an unbiased ELBO flips the ordering of active learning performance for Monte Carlo dropout and Radial BNNs. Radial BNNs (solid line) perform best when a biased estimator is used (blue) but worst with the unbiased estimator (orange). MCDO (dashed line) performs in the middle with both biased and unbiased estimators. Because the ordering of the performance evaluations of methods is sensitive to de-biasing, active learning is not a reliable way to evaluate the approximate inference. Shading represents standard error. \tilde{R}_{PURE} is omitted for visual clarity but is very similar to the result for \tilde{R}_{LURE} and is shown in fig. C.3.

At the same time, for Fashion MNIST, the differences between the two approximate posteriors are large enough that the effect of de-biasing does not change the ordering. This highlights that Bayesian active learning performance is not insensitive to the choice of posterior approximation, it just does not offer a definitive ordering of methods.

Empirical Investigation of Bias in Active Model Evaluation

This is in contrast with active model evaluation—in which we attempt to estimate the test loss of a *fixed* model. We take a fixed and fully trained model and then actively acquire labels from the test dataset. Our goal is to estimate the full test risk with as few datapoints as possible. The overall approach is discussed in more detail by Kossen et al. [2021].

In this case, it is clear that removing the bias is the only sensible way to interpret the task—the goal of model evaluation is to discover the actual risk of the model, the evaluation is not being used to implicitly guide the model towards informative data. Now, the only impact of an improved approximate posterior distribution is to better estimate the mutual information between the parameters and data labels. The more quickly the variance of the test risk estimator reduces, the better the approximate posterior distribution.

We can measure performance using, for example, the mean-squared error between the actively-sampled risk estimate and the full test-pool risk (recall that we do not expect to be able to fix the gap between the full test pool and the true population risk using active sampling alone).

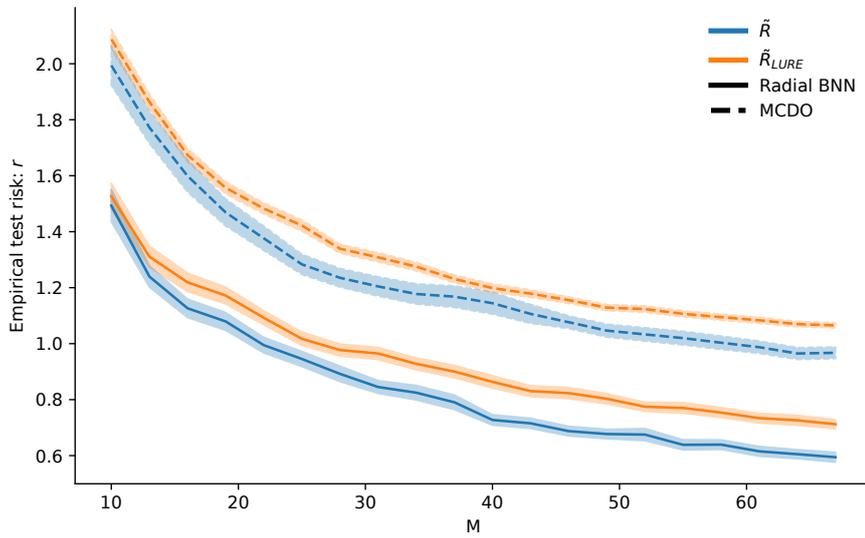
In fig. 5.7, we show the mean-squared error of the actively sampled risk estimator for both MNIST and FashionMNIST as a way of comparing Radial BNNs and Monte Carlo dropout. In both cases, Radial BNNs appear to be providing much better guidance as to the expected information gain provided by test labels. Although we only examine small image datasets here, [Kossen et al., 2021] explores active testing in much larger settings, including CIFAR-100, in work which extends beyond the scope of this thesis.²

I also want to emphasise that although active testing avoids *one* of the problems of using active learning as a measure of approximate posterior performance, I have not shown that it does not suffer from other problems of its own.

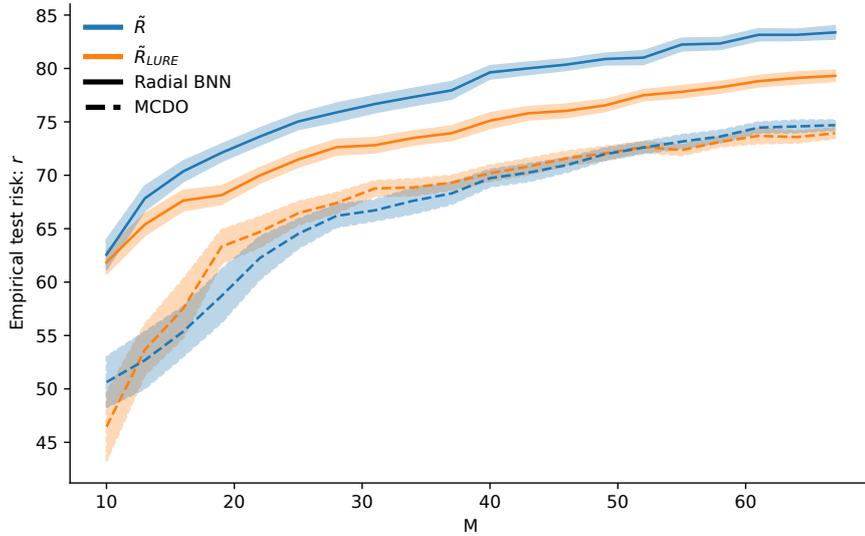
Active Learning Bias in the Context of Overall Bias

In the previous sections, we relied on the observation that removing the active learning bias can hurt active learning performance. This is counterintuitive, so it would help to have a plausible explanation of this phenomenon in order to reassure ourselves that we understand roughly what is happening. We

²The author of this thesis is a joint-first-author of Kossen et al. [2021] but anticipates that his co-author will submit some or all of that paper in their own thesis.

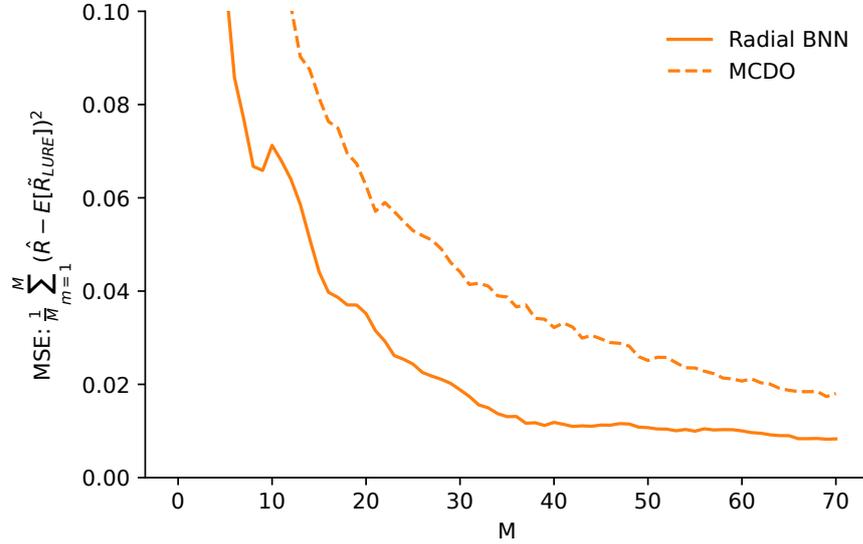


(a) Test loss.

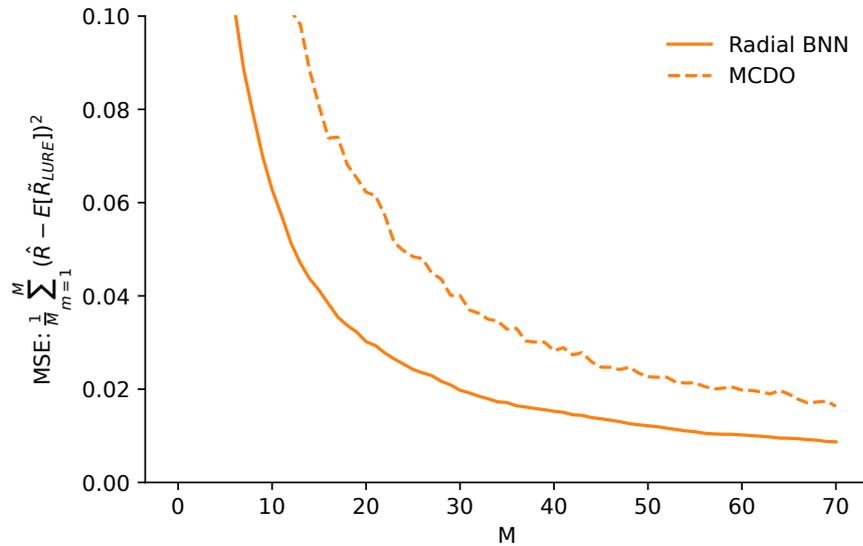


(b) Test accuracy.

Figure 5.6: **FashionMNIST**. In some cases, the effect of debiasing is small enough to allow active learning to reveal performance differences. Here, MCDO performs sufficiently worse than Radial BNNs that even though the debiasing step hurts active learning performance more for Radial BNNs, the ordering is preserved.

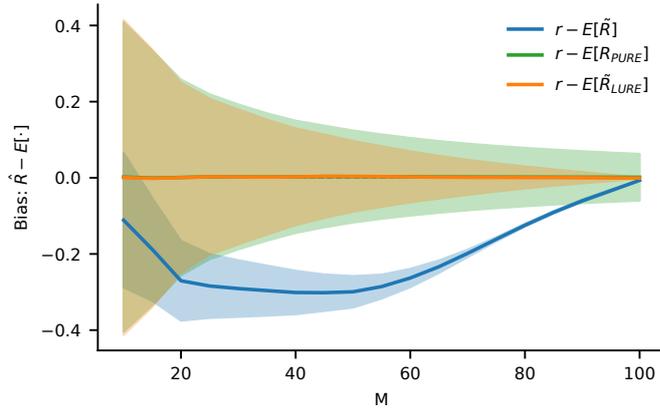


(a) MNIST.

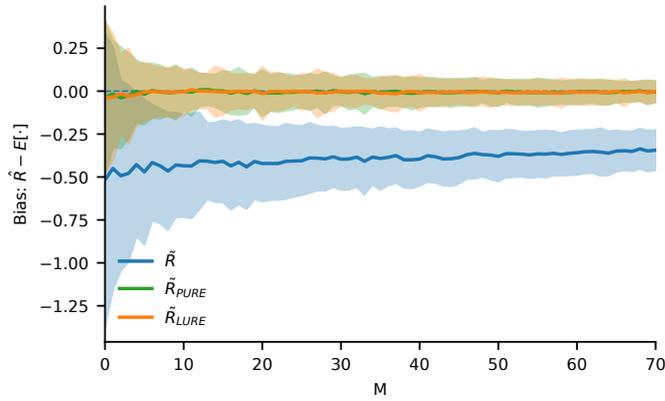


(b) FashionMNIST.

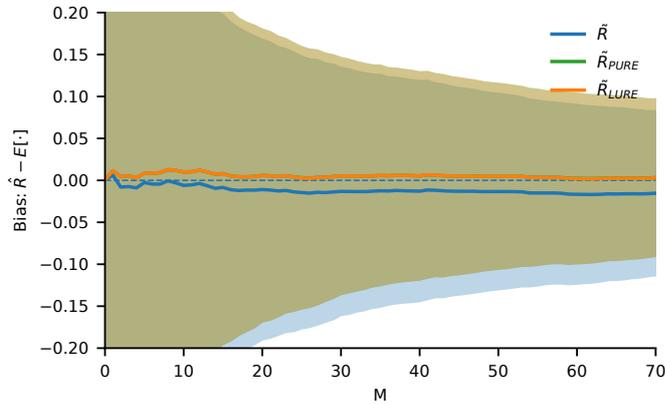
Figure 5.7: **Active Evaluation Mean-squared Error.** We compute the error between the actively evaluated risk and the ‘true’ population risk (as evaluated on the full test dataset). Unlike active *learning*, the unbiased risk estimate is genuinely what we value, not a means to an end, which makes this a more effective evaluation of the approximate posterior. These specific evaluations suggest that the Radial BNN is serving as a more effective approximate posterior than MCDO in this case.



(a) Linear regression



(b) Radial: MNIST



(c) Radial: Fashion

Figure 5.8: \tilde{R}_{PURE} and \tilde{R}_{LURE} remove a bias introduced by active learning, while unweighted \tilde{R} , which most active learning work uses, is biased. Note the sign: \tilde{R} *overestimates* risk because active learning samples the hardest points. Variance for \tilde{R}_{PURE} and \tilde{R}_{LURE} depends on the acquisition distribution placing high weight on high-expected-loss points, which makes this an evaluation of the approximate posterior distribution. Shading is ± 1 standard deviation.

hypothesise that the finding that \tilde{R}_{LURE} hurts training for the BNN makes sense in the context of the bias introduced by overfitting. That is to say, we need to examine the effect of removing statistical bias in the context of *overall* bias—training would ordinarily induce an overfitting bias (OFB) even if we had not used active learning.

If we optimize parameters θ according to \hat{R} , then $\mathbb{E} \left[\hat{R}(\theta^*) \right] \neq r$, because the optimized parameters θ^* tend to explain training data better than unseen data. Using \tilde{R}_{LURE} , which removes *statistical bias*, we can isolate OFB in an active learning setting. More formally, supposing we are optimizing any of the discussed risk estimators (which we will write using $\tilde{R}_{(\cdot)}$ as a placeholder to stand for any of them) we define the OFB as:

$$B_{\text{OFB}}(\tilde{R}_{(\cdot)}) = r - \tilde{R}_{\text{LURE}}(\theta^*) \quad \text{where} \quad \theta^* = \arg \min_{\theta}(\tilde{R}_{(\cdot)}) \quad (5.10)$$

$B_{\text{OFB}}(\tilde{R}_{(\cdot)})$ depends on the details of the optimization algorithm and the dataset. Understanding it fully means understanding generalization in machine learning and is outside our scope. We can still gain insight into the interaction of active learning bias (ALB) and OFB. Consider the possible relationships between the magnitudes of ALB and OFB:

ALB \gg OFB Removing ALB reduces overall bias and is most likely to occur when f_{θ} is not very expressive such that there is little chance of overfitting.

ALB \ll OFB Removing ALB is irrelevant as model has massively overfit regardless.

ALB \approx OFB Here *sign* is critical. If ALB and OFB have opposite signs and similar scale, they will tend to cancel each other out. Indeed, they usually have opposite signs.

B_{OFB} is usually positive: θ^* fits the training data better than unseen data. ALB is generally negative: we actively choose unusual, surprising, or informative points which are harder to fit than typical points.

Therefore, when significant overfitting is possible, unless ALB is also large, removing ALB will have little effect and can *even be harmful*. This hypothesis would explain the observations in §5.2 if we were to show that B_{OFB} was small for linear regression but had a similar magnitude and opposite sign to ALB for the BNN. This is exactly what we show in Figure 5.9.

Specifically, we see that for linear regression, the B_{OFB} for models trained with \tilde{R} , \tilde{R}_{PURE} , and \tilde{R}_{LURE} are all small (Figure 5.9a) when contrasted to the ALB shown in Figure 5.8a. Here ALB \gg OFB; removing ALB matters. For BNNs we instead see that the OFB has opposite sign to the ALB but is either similar in scale for MNIST (Figures 5.8b and 5.9b), or the OFB is much larger than ALB for Fashion MNIST (Figures 5.9c and 5.8c). The two sources of bias

thus (partially) cancel out. Essentially, using active learning can be treated (quite instrumentally) as an ad hoc form of regularization. This explains why removing ALB can hurt active learning with neural networks.

Related Work on Unbiased Active Learning

There have been some attempts to address active learning bias, but these have generally required fundamental changes to the active learning approach and only apply to particular setups. Beygelzimer et al. [2009], Chu et al. [2011], and [Cortes et al., 2019] apply importance-sampling corrections [Sugiyama and Jp, 2006, Bach, 2006] to *online* active learning. Unlike pool-based active learning, this involves deciding whether or not to sample a new point as it arrives from an infinite distribution. This makes importance-sampling estimators much easier to develop, but as Settles [2010] notes, “*the pool-based scenario appears to be much more common among application papers.*”

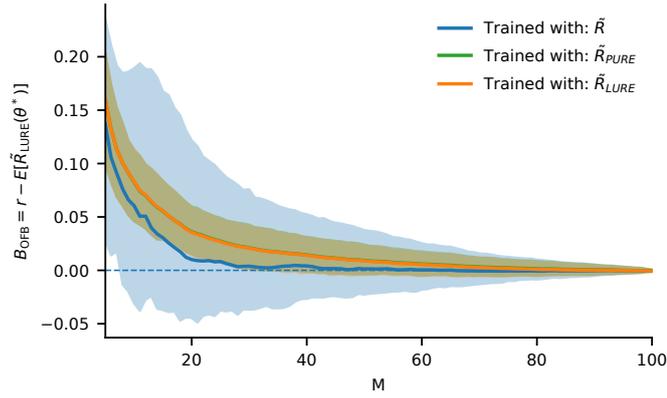
Ganti and Gray [2012] address unbiased active learning in a pool-based setting by sampling from the pool *with replacement*. This effectively converts pool-based learning into a stationary online learning setting, although it overweights data that happens to be sampled early. Sampling with replacement is unwanted in active learning because it requires retraining the model on duplicate data which is either impossible or wasteful depending on details of the setting. Moreover, they only prove the consistency of their estimator under very strong assumptions (well-specified linear models with noiseless labels and a mean-squared-error loss). Imberg et al. [2020] consider optimal proposal distributions in an importance-sampling setting. Outside the context of active learning, Byrd and Lipton [2019] question the value of importance-weighting for deep learning, which aligns with our findings below.

5.3 Conclusion

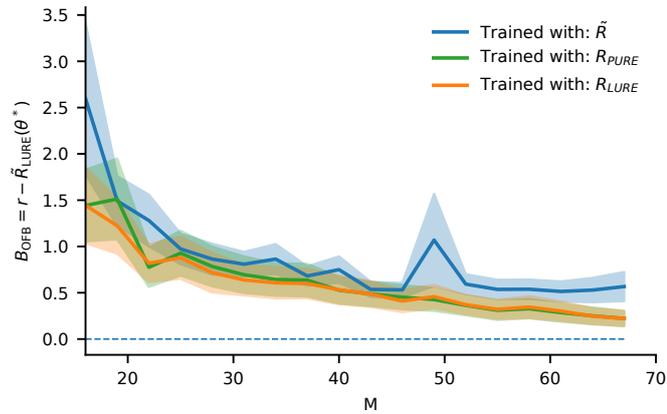
Continual and active learning can both be used to test interesting parts of Bayesian approximate posterior distributions. In both cases, there are perfectly valid non-Bayesian approaches to solving the problem. But when a Bayesian approach is taken, performance on the underlying task can be used as an evaluation of the degree to which the Bayesian approximation is succeeding in its goals of updating sequentially or representing uncertainty.

However, we have shown that continual learning evaluations must be carefully constructed to show that it is the Bayesian sequential updating which is doing the ‘heavy lifting’, rather than relying on cached information. When constructed in a rigorous way many standard approximate inference schemes struggle greatly on even very simple tasks.

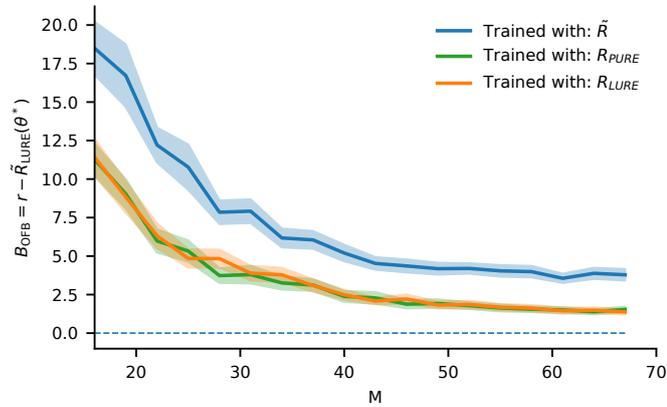
Active learning, in contrast, may just be the wrong problem. We are accidentally evaluating the ability of the approximate posterior to inject implicit biases into the data-gathering which represents prior assumptions which we



(a) Linear regression



(b) Radial: MNIST



(c) Radial: Fashion

Figure 5.9: Overfitting bias— B_{OFB} —for models trained using the three objectives. (a) Linear regression, B_{OFB} is small compared to ALB (c.f. Figure 5.8a). Shading IQR. 1000 trajectories. (b) Radial BNN, B_{OFB} is similar scale and opposite magnitude to ALB (c.f. Figure 5.8b). (A single outlier 2 orders of magnitude greater than the rest caused the spike at 50 in this figure.) (c) Radial BNN on FashionMNIST, OFB is somewhat larger than with MNIST, particularly for \tilde{R} (i.e. our approaches reduce overfitting) and dominates active learning bias (c.f. Figure 5.8c). Shading ± 1 standard error. 150 trajectories.

neither understand nor can specify. Applying a de-biasing trick can reverse the ordering of model performance indicated by doing active learning. Perhaps, active model evaluation may be a more suitable evaluation for approximate posterior distributions because it avoids some unintended interactions between model parameters and data which are present in active learning.

However, a major limitation of this entire approach is that it is hard to know how general any lessons learned from these evaluations can be. If we find that one approximate inference technique performs very well on a well-constructed continual learning evaluation with some dataset, should we expect it to perform well on a different dataset? On a differently constructed continual learning evaluation? Of course, we know that predicting this is a hard problem, which receives enormous amounts of attention in the field of machine learning.

Using this evaluation strategy, then, have we really said anything about the underlying approximate inference? Or are we only able to make much more limited claims about specific problems?

I think the truth is somewhere in between. The more generalizable the good performance is (across a wide range of specific but similar evaluations) the more confident we can be that it is something about the underlying technique which is producing those results in a generally effective way. But we will never be able to say with complete confidence, on this approach, that one Bayesian approximation is superior to another (in the way that we might have hoped following the intuitions of Dutch books, for example).

Chapter 6

Conclusion and Discussion

We set out hoping to motivate a principled automatic system for reasoning about beliefs given evidence. Bayesian motivations for the fundamental role of belief distributions led us towards intractable Bayesian inference [Ramsey, 1926, de Finetti, 1937, Cox, 1961, Jaynes, 2003]. Approximations have a long history of success in science as a way of overcoming intractable but principled calculations. But the original, historical, motivations for Bayesian inference were binary. Under the overly-demanding requirements set by those early motivations for Bayesian inference, a set of belief distributions is either ‘rational’ or ‘irrational’ depending on whether or not they follow the rules of probability. Anything that is approximately Bayesian is, within this framework, more properly *not* Bayesian, and irrational. This includes both approximations in model choice and computational approximations. As a result, it appears that no physically instantiable reasoning system (and certainly no person) is ‘rational’ under these overly-demanding requirements.

This leaves us with the central organizing question of this thesis: is there a good way of deciding when an approximately Bayesian method is doing a good job of acting like a Bayesian tool? In short: how can we evaluate the success of a Bayesian approximation?

One solution which is clearly not good enough is to use the same evaluations that every other method uses. In principle, Bayesian methods ought to unlock entirely new *machine learning behaviours*. While Bayesian methods might (or might not) allow better generalization on standard predictive tasks, that should not be their main attraction. Being able to treat your machine learning platform as a distribution, not just a single estimate, allows you to adapt your entire pipeline in new ways, for example by actively guiding labelling or exploration activity.

We briefly considered using a distance metric or divergence between an approximate and ‘true’ posterior to establish an ordering of more and less Bayesian methods. Investigating this quickly revealed a problem: not only was it challenging to pick a preferred metric, but plausible candidate metrics disagreed about ordering. Moreover, we were able to show that the ‘cost’ imposed by various approximations is intimately tied up with details of the situation. The same approximation algorithm can have very different results depending on the parametric function, approximating family, dataset, and optimization algorithm. For example, we showed in chapter 3 that the commonly-used mean-field assumption can be extremely restrictive for small models, but may be much less of a constraint for some purposes in larger parametric models whose connections can re-introduce the desired complexity in predictive distributions. Similarly, we showed in chapter 4 that the choice of approximating distribution can interact with the use of gradient-based optimizers by affecting the sampling properties of stochastic gradient estimators.

It also appears to be difficult to adapt the basic structure of the historical, more ‘binary’, arguments motivating Bayesian inference to a measure of ‘Bayesianness’ or degree-of-rationality. Arguments based on axioms in the style of Cox [1961] and Jaynes [2003] are difficult to express in a ‘soft’ form. Axioms are either tight enough to constrain us to require our beliefs to be manipulated like probabilities or they are not. Without a deep conceptual breakthrough of the kind I was not able to discover, it is not clear what sort of pseudo-probability would be strong enough that in the limit of infinite resources becomes perfectly rational and which corresponds to a measure of degrees-of-rationality that allows us to compare two inferences to decide which we ought to prefer.

Similarly, methods that are based on Dutch-books seem to me to be resistant to being made ‘soft’. The moment you allow *some* departure from the rational ideal, one’s belief distribution is definitely exploitable. To turn a Dutch-book argument into something that can measure degrees-of-rationality we would need to add concepts for degree-of-exploitability. My best effort to define these degrees returns again and again to *rates* of loss. But the actual expected rate of loss depends not on the difference between your own approximation and the true posterior—rather it depends on the interaction between your approximation, the *actual data-distribution*, and your utility function. At the same time, your maximal rate of loss depends just on the one point in belief-space where your mistake causes the largest possible penalty (in turn this depends on your utility function). Maximal rate of loss is not sensitive to the rest of your distribution, nor does the Bayesian ideal minimize the maximum rate of loss. In short, my attempts to soften Dutch-book arguments to define degrees-of-exploitability wound up in the same pseudo-frequentist space as the expected utility measures which I ended up considering.

It seems hard, therefore, to rescue our evaluation of degrees-of-Bayesianness by adapting the original binary motivations for Bayesian inference. There are,

however, attempts to do something similar from the fields of bounded rationality and probabilistic numerics.

Bounded rationality-style approaches [Wheeler, 2020] offer some mechanisms for understanding the necessity of ‘imperfect’ reasoning due to the importance of approximations and heuristics. Bounded rationality research has also explored how these sorts of approximations occur in humans both within psychology and economics. However, less emphasis has been placed on principled reasoning under bounded modelling and computation, and a key question which remains unresolved is how to know which of two approximations is ‘better’ in some context. It seems unlikely to me that any such answer can be grounded without reference to expected data-distributions and decision-problems which is why I turn to expected utility of approximate distributions.

Relatedly, probabilistic numerics [Hennig et al., 2022] frames computational approximation as an inference problem in its own right. When we are trying to estimate some quantity, we can compute it probabilistically and understand the computation as attempting to minimize our uncertainty about the answer given some degree of effort. However, there are two reasons why this approach does not quite solve our problems. First, probabilistic numerics broadly describes how to apply probabilistic tools to the problem of numerical estimation, but that does not mean that it provides a way of framing arbitrary numerical methods as probabilistic ones. That is to say, given some apparently successful numerical method (or heuristic algorithm) it is generally not obvious what probabilistic assumptions correspond to that heuristic. An example of this can be seen in chapter 3 and chapter 4. In those chapters, we explored how variational methods relied on pragmatic heuristics that made their Bayesian interpretation very complicated indeed. This means that probabilistic numerics is almost solving the inverse problem to the one which this thesis focuses on: using Bayesian methods to do numerics rather than understanding what numerical tricks pragmatically-Bayesian methods are actually depending on. Second, existing approaches to probabilistic numerics broadly do not answer the issue of which modelling assumptions and inference approximations to use as part of the computation effort. By applying the Bayesian toolkit to the computational procedure we give ourselves all the same tools for model selection that we originally had for Bayesian inference, but no new ones. In this way, the question of evaluation (which method is ‘better’) is unchanged by the use of probabilistic numerics. We still either need some ‘soft’ but purely Bayesian criterion of ‘correct reasoning’ or we need to resort to semi-frequentist tools like expected utility.

This inherent case-by-case nature of the quality of Bayesian approximations led us to a more concrete way of understanding approximation quality. Within a particular *decision-problem*—in the context of a given data-distribution and utility function—we can evaluate the expected utility of an approximate posterior distribution.

At its simplest, we could imagine picking a single universal decision-problem and using it as a proxy for quality generally [Key et al., 1999]. Although this might seem like a hopelessly naive approach, it is important to remember that much of current machine learning more-or-less does exactly this! The procedure of comparing computer-vision architectures based on their top- k ImageNet accuracy, for example, is precisely identical with that approach. We can make this very naive approach richer by increasing the range of performance measures which are considered as well as the variety of decision-problems. At the furthest opposite extreme, we could say that the notion of general quality is nonsense, and each approximating distribution needs to be judged entirely fresh on each new problem.

This thesis can be thought of as taking some steps to flesh out two possible families of decision-problems as possible evaluations. A good evaluation should be:

- Relevant: good performance on an exemplar of the decision-problem family ought to be useful.
- Representative: good performance on an exemplar of the decision-problem family ought to correlate with good performance on all members of the family.
- Well-posed: the boundaries of the decision-problem family should be sufficiently clearly specified that practitioners can generally spot when the appropriate evaluation is being used.
- Isolated: the decision-problem ought, as much as possible, to test an isolated property of the model being investigated without muddling several mechanisms of action together so that it is hard to know which is causing success or failure.

We can select both continual and active learning as decision-problem families that might be suitable for evaluations of approximate Bayesian inference. Bayesianism characterized by the idea that beliefs about the world ought to be represented by probability distributions. Continual learning is an example of a decision-problem that checks if the sequential updating process in an approximate Bayesian method is working. If a method is trying to perform sequential updating but is not succeeding at continual learning, then it is not actually performing sequential probabilistic inference. Meanwhile, active learning is an example of a decision-problem that checks if the distributions represent beliefs about the world. If active learning methods based on the expected information gain about those belief distributions are working well then it shows that the belief distributions are reflecting the actual epistemic state with respect to the data and model.

In both cases, I believe that the decision-problems are relevant. They are fairly representative, but not perfectly. In both cases, good performance in

one setting can be indicative of good general performance on that style of decision-problem. Continual learning methods’ performances can depend on the dataset (witness, for example, the special features of the permuted MNIST dataset for continual learning which are discussed in chapter 5). Meanwhile, active learning methods can work well on specific kinds of data without performing well on others (for example, the failures of entropy-based methods to work well for data with heteroskedastic aleatoric noise).

Both active and continual learning are relatively well-posed, at least sufficiently for sub-fields to arise to address each (even if, by my recollections at least, the panel of the 2018 continual learning workshop at ICML agreed to disagree on what the aim of the field was!). I also chose these two decision-problem families partly because I think they are relatively isolated. It is entirely possible to build an approximate inference method that performs extremely well on some measures of fitting the underlying dataset but does very badly at either active or continual learning. For example, a Bayesian neural network fit with variational inference can trivially get near-perfect accuracy on MNIST but can still not carry out single-headed Split MNIST with better than chance catastrophic forgetting. This suggests that there is something separable about the continual learning and prediction tasks. Evaluations based on this characteristic behaviour can therefore start to tease apart different sorts of failures and successes of Bayesian inference.

We must be clear that performing well on these problems is *not*, on its own, evidence that a method is doing good Bayesian inference. On the contrary, it is possible to construct heuristic approaches to these problems which do not rely on any Bayesian fundamentals at all. For example, one possible solution to continual learning is simply to allow oneself extra parameters for each new task and then to stitch things together at the end [Schwarz et al., 2018]. Rather, if a method is constructed to perform approximate Bayesian inference as a solution to these problems but a well-designed evaluation shows that they are performing badly then it is evidence that the approximate inference is not working as intended.

As part of this investigation, however, we came across some ways in which evaluations based on continual or active learning are not optimal. Continual learning evaluations can be made too easy to solve, and multi-headed versions of the problem add extra parameters that lie outside the motivating Bayesian formalism. In addition, several methods that are explicitly Bayesian in fact rely heavily on non-Bayesian components that smuggle information from the past via side-channels which lie outside of the approximate inference step. At the same time, sufficiently rigorous evaluations reveal the simple fact that existing Bayesian approximations in parameter-space are not doing a good enough job capturing the parameter distribution to allow sequential updating.

Active learning, meanwhile, is maybe not quite isolated enough. We showed that the goal of picking informative points was interacting with some effective heuristics which act as implicit regularization. While there is nothing

wrong with performing implicit regularization, because it lies outside the explicit probabilistic formalism of the problem definition, the importance of the implicit inductive bias means that active learning is an unreliable evaluation for the effectiveness of the underlying approximate inference.

Instead of either of these, we propose two scenarios that appear to be good evaluations of important parts of Bayesian inference. We see approximate Bayesian inference as chiefly characterized by the idea that distributions over parameters reflect subjectively uncertain beliefs about the states they represent. In this context, continual learning and active learning are especially relevant. The former tests the ability of the current belief representation to encode the evidence accumulated so far. The latter tests the ability of the current belief representation to reflect missing information.

We considered whether active *testing* [Kossen et al., 2021] might be an alternative evaluation for approximate inference. In the case of active testing, because the model-performance itself does not depend on the implicit bias, removing the bias might be just the ‘right’ thing to do, providing a clearer evaluation metric. In fact, however, it is not necessarily the case that this is better. After all, a biased but lower-variance testing regime might be better for some utility functions, making even active testing a possibly non-isolated evaluation for approximate Bayesian inference.

Despite the challenges facing both continual and active learning as evaluations of Bayesian inference, I nevertheless believe that these are an excellent place to start. Future work ought to develop more, and more careful, evaluations that test the characteristic behaviours of Bayesian inference in a variety of ways that collectively allow us to build a picture of the successes and failures of different approximation methods in different settings.

We should also always bear in mind that these evaluations cannot be blindly applied, because testing the quality of the Bayesian approximation depends on the evaluation being set up in a way that *actually* tests that. It would be easy, for example, to design a so-called Bayesian approximation with the problem of active learning specifically in mind which smuggled some geometric heuristics under the hood and performed very well on some example problem where those heuristics happened to be effective. At the same time, an approximation which performs extremely well for one dataset or architecture might genuinely perform badly on another because of unexpected interactions between the underlying components of the approximation and learning procedure.

An evaluation is not quite the same thing as a benchmark, and I suspect that the best ways to identify the ways in which Bayesian approximations are and are not working might not lend themselves very well to benchmark construction because they might be easy to fake. As scientists, however, we can rise to the challenge of probing our models fairly and carefully and trying to genuinely understand which ones are working and why.

Bibliography

- Pierre Alquier. Approximate Bayesian Inference. *Entropy*, 22(1272), 2020. doi: 10.3390/e22111272.
- Les Atlas, David Cohn, and Richard Ladner. Training Connectionist Networks with Queries and Selective Sampling. *Neural Information Processing Systems*, 1990.
- Francis R. Bach. Active learning for misspecified generalized linear models. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 65–72, Cambridge, MA, USA, December 2006. MIT Press.
- David Barber and Christopher M Bishop. Ensemble Learning in Bayesian Neural Networks. In *Neural Networks and Machine Learning*, pages 215–237. Springer, 1998.
- William H Beluch, Tim Genewein, Andreas Nurnberger, and Jan M Kohler. The Power of Ensembles for Active Learning in Image Classification. *CVPR*, 2018.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, pages 41–48, New York, NY, USA, June 2009. Association for Computing Machinery. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553380.
- Michael Betancourt. A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv:1701.02434 [stat]*, July 2018.
- Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 49–56. Association for Computing Machinery, New York, NY, USA, June 2009. ISBN 978-1-60558-516-1.

- Chris Bishop. Introduction. In *Pattern Recognition and Machine Learning*. Springer, 2006.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. *Proceedings of the 32nd International Conference on Machine Learning*, 37:1613–1622, 2015.
- Lennart Bondesson. A Class of Probability Distributions that is Closed with Respect to Addition as Well as Multiplication of Independent Random Variables. *Journal of Theoretical Probability*, 28(3):1063–1081, September 2015.
- George E. P. Box and Mervin E. Muller. A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.
- Wray L. Buntine and Andreas S. Weigend. Bayesian Back-Propagation. *Complex Systems*, 5(6):603–643, 1991.
- Jonathon Byrd and Zachary C. Lipton. What is the Effect of Importance Weighting in Deep Learning? *ICML*, June 2019.
- Rich Caruana. Multitask Learning. *Machine Learning*, 28(1):41–75, July 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734.
- Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. *ECCV*, 2018.
- Jiaoyan Chen, Yan Zhou, Alexander Zipf, and Hongchao Fan. Deep Learning From Multiple Crowds: A Case Study of Humanitarian Mapping. *IEEE Transactions on Geoscience and Remote Sensing*, 57(3):1713–1722, March 2019. ISSN 1558-0644. doi: 10.1109/TGRS.2018.2868748.
- Tianqi Chen, Emily B Fox, and Carlos Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. *ICML*, 2014.
- Min Chee Choy, Dipti Srinivasan, and Ruey Long Cheu. Neural Networks for Continuous Online Learning and Control. *IEEE Transactions on Neural Networks*, 17(6):1511–1531, November 2006. ISSN 1941-0093. doi: 10.1109/TNN.2006.881710.
- Wei Chu, Martin Zinkevich, Lihong Li, Achint Thomas, and Belle Tseng. Unbiased online active learning in data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 195–203, New York, NY, USA, August 2011. Association for Computing Machinery. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020444.

- Adam D. Cobb and Brian Jalaian. Scaling Hamiltonian Monte Carlo inference for Bayesian neural networks with symmetric splitting. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 675–685. PMLR, December 2021.
- Adam D. Cobb, Stephen J. Roberts, and Yarin Gal. Loss-Calibrated Approximate Inference in Bayesian Neural Networks, May 2018.
- Adam D Cobb, Atılım Günes Baydin, Ivan Kiskin, Andrew Markham, and Stephen J Roberts. Semi-separable Hamiltonian Monte Carlo for inference in Bayesian neural networks. *preprint*, 2019a.
- Adam D. Cobb, Atılım Güneş Baydin, Andrew Markham, and Stephen J. Roberts. Introducing an Explicit Symplectic Integration Scheme for Riemannian Manifold Hamiltonian Monte Carlo. *arXiv*, October 2019b.
- Corinna Cortes, Giulia Desalvo, Mehryar Mohri, Ningshan Zhang, and Claudio Gentile. Active Learning with Disagreement Graphs. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1379–1387. PMLR, May 2019.
- Richard Cox. *The Algebra of Probable Inference*. 1961.
- Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning - ICML '08*, pages 208–215, Helsinki, Finland, 2008. ACM Press. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390183.
- Bruno de Finetti. Foresight: Its Logical Laws, Its Subjective Sources. In *Studies in Subjective Probability*. Robert E. Krieger Publishing Co, Huntington, NY, 1937.
- John S. Denker and Yann leCun. Transforming Neural-Net Output Levels to Probability Distributions. *Neural Information Processing Systems*, pages 853–859, 1991.
- J. L. Doob. Application of the theory of martingales. *Le Calcul des Probabilites et ses Applications*, pages 23–27, 1949.
- Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer, 2001.
- Sebastian Farquhar and Yarin Gal. Differentially Private Continual Learning. *Privacy in Machine Learning and AI workshop at ICML*, 2018a.
- Sebastian Farquhar and Yarin Gal. A Unifying Bayesian View of Continual Learning. *Bayesian Deep Learning Workshop at NeurIPS*, 2018b.

- Sebastian Farquhar, Michael Osborne, and Yarin Gal. Radial Bayesian Neural Networks: Robust Variational Inference In Big Models. *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, 2020.
- Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On Statistical Bias In Active Learning: How and When to Fix It. *International Conference on Learning Representations*, 2021.
- V Fedorov. *Theory Of Optimal Experiments*. Academic Press, New York, 1972. ISBN 978-0-323-16246-3.
- Angelos Filos, Sebastian Farquhar, Aidan N Gomez, Tim G J Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. Benchmarking Bayesian Deep Learning with Diabetic Retinopathy Diagnosis. *Bayesian Deep Learning Workshop at NeurIPS*, 2019.
- Andrew Y K Foong, David R Burt, Yingzhen Li, and Richard E Turner. On the Expressiveness of Approximate Inference in Bayesian Neural Networks. *Advances In Neural Information Processing Systems*, 2020.
- Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian Recurrent Neural Networks. *arXiv preprint*, 2017.
- Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy Networks for Exploration. *ICLR*, 2018.
- Yarin Gal. *Uncertainty in Deep Learning*. DPhil, University of Cambridge, 2016.
- Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *Proceedings of the 33rd International Conference on Machine Learning*, 48:1050–1059, 2015.
- Yarin Gal, Jiri Hron, and Alex Kendall. Concrete Dropout. *Neural Information Processing Systems*, page 10, 2017a.
- Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian Active Learning with Image Data. *Proceedings of The 34th International Conference on Machine Learning*, 2017b.
- Ravi Ganti and Alexander Gray. UPAL: Unbiased Pool Based Active Learning. *Artificial Intelligence and Statistics*, 2012.
- Zoubin Ghahramani. Should all Machine Learning be Bayesian? Should all Bayesian models be non-parametric? In *Bayesian Research Kitchen Workshop*, Grasmere, October 2008.

- Sambuddha Ghosal, Bangyou Zheng, Scott C. Chapman, Andries B. Potgieter, David R. Jordan, Xuemin Wang, Asheesh K. Singh, Arti Singh, Masayuki Hirafuji, Seishi Ninomiya, Baskar Ganapathysubramanian, Soumik Sarkar, and Wei Guo. A Weakly Supervised Deep Learning Framework for Sorghum Head Detection and Counting. <https://spj.sciencemag.org/journals/plantphenomics/2019/1525874/>, June 2019.
- Micah Goldblum, Jonas Geiping, Avi Schwarzschild, Michael Moeller, and Tom Goldstein. Truth or backpropaganda? An empirical investigation of deep learning theory. In *International Conference on Learning Representations*, September 2019.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv*, 2013.
- Alex Graves. Practical Variational Inference for Neural Networks. *Neural Information Processing Systems*, 2011.
- Robert M. Gray. *Entropy and Information Theory*. Springer US, second edition, 2011. ISBN 978-1-4419-7969-8. doi: 10.1007/978-1-4419-7970-4.
- Peter Grunwald. Safe Learning: Bridging the gap between Bayes, MDL and statistical learning theory via empirical convexity. *Conference on Learning Theory*, 2011.
- Juan Mario Haut, Mercedes E. Paoletti, Javier Plaza, Jun Li, and Antonio Plaza. Active Learning With Convolutional Neural Networks for Hyperspectral Image Classification Using a New Bayesian Approach. *IEEE Transactions on Geoscience and Remote Sensing*, 56(11):6440–6461, November 2018. ISSN 1558-0644. doi: 10.1109/TGRS.2018.2838665.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 7(3):171–180, 2016.
- Philipp Hennig, Michael A. Osborne, and Hans P. Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, Cambridge, 2022. ISBN 978-1-107-16344-7. doi: 10.1017/97811316681411.
- José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

- José Miguel Hernández-Lobato, Yingzhen Li, Mark Rowland, Daniel Hernández-Lobato, Thang Bui, and Richard E. Turner. Black-box α -divergence Minimization. *International Conference on Machine Learning*, June 2016.
- Geoffrey Hinton and Drew van Camp. Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, 1993.
- Matthew D Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014.
- Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian Active Learning for Classification and Preference Learning. *arXiv:1112.5745 [cs, stat]*, December 2011.
- Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming Catastrophic Forgetting via Model Adaptation. *International Conference on Learning Representations*, 2019.
- Sheng-Jun Huang, Jia-Wei Zhao, and Zhao-Yang Liu. Cost-Effective Training of Deep CNNs with Active Model Adaptation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18*, pages 1580–1588, New York, NY, USA, July 2018. Association for Computing Machinery. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3220026.
- Ferenc Huszar. Note on the quadratic penalties in elastic weight consolidation. *PNAS*, 115(11):E2496–E2497, 2018.
- Henrik Imberg, Johan Jonasson, and Marina Axelson-Fisk. Optimal sampling in unbiased active learning. *AI Stats*, 2020.
- Pavel Izmailov, Sharad Vikram, Matthew D. Hoffman, and Andrew Gordon Wilson. What Are Bayesian Neural Network Posteriors Really Like? *International Conference on Machine Learning*, April 2021.
- Tommi S. Jaakkola and Michael I. Jordan. Improving the Mean Field Approximation Via the Use of Mixture Distributions. In Michael I. Jordan, editor, *Learning in Graphical Models*, NATO ASI Series, pages 163–173. Springer Netherlands, Dordrecht, 1998. doi: 10.1007/978-94-011-5014-9_6.
- Frank Jackson and Robert Pargetter. A Modified Dutch Book Argument. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, 29(6):403–407, 1976. ISSN 0031-8116.

- Edwin Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, April 2003. ISBN 0-521-59271-2.
- Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.
- Pasi Jylänki, Aapo Nummenmaa, and Aki Vehtari. Expectation Propagation for Neural Networks with Sparsity-Promoting Priors. *Journal of Machine Learning Research*, 15(54):1849–1901, 2014. ISSN 1533-7928.
- Kaggle. Diabetic Retinopathy Detection, 2015.
- H. Kahn and A. W. Marshall. Methods of Reducing Sample Size in Monte Carlo Computations. *Journal of the Operations Research Society of America*, 1(5):263–278, November 1953. ISSN 0096-3984. doi: 10.1287/opre.1.5.263.
- Benjamin Kellenberger, Diego Marcos, Sylvain Lobry, and Devis Tuia. Half a Percent of Labels is Enough: Efficient Animal Detection in UAV Imagery Using Deep CNNs and Active Learning. *IEEE Transactions on Geoscience and Remote Sensing*, 57(12):9524–9533, December 2019. ISSN 0196-2892. doi: 10.1109/TGRS.2019.2927393.
- John G. Kemeny. Fair Bets and Inductive Probabilities. *The Journal of Symbolic Logic*, 20(3):263–273, 1955. ISSN 0022-4812. doi: 10.2307/2268222.
- Samuel Kessler, Vu Nguyen, Stefan Zohren, and Stephen J. Roberts. Hierarchical Indian buffet neural networks for Bayesian continual learning. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 749–759. PMLR, December 2021.
- Jane T. Key, Luis R. Pericchi, and Adrian F. M. Smith. Bayesian Model Choice: What and Why? *Bayesian Statistics*, 6:343–370, 1999.
- Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. *International Conference on Machine Learning*, 2018.
- Diederik P. Kingma, Maz Welling, and Soumith Chintala. Auto-Encoding Variational Bayes. *International Conference on Learning Representations*, 2014.
- Diederik P. Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. *Advances In Neural Information Processing Systems*, pages 2575–2583, 2015.

- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, March 2017.
- Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. *arXiv:1906.08158 [cs, stat]*, June 2019.
- B. J. K. Kleijn and A. W. van der Vaart. The Bernstein-Von-Mises theorem under misspecification. *Electronic Journal of Statistics*, 6(none):354–381, January 2012. ISSN 1935-7524, 1935-7524. doi: 10.1214/12-EJS675.
- Andrey Kolmogorov. *Foundations of the Theory of Probability*. Chelsea Publishing Company, New York, first english translation edition, 1950.
- Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active Testing: Sample-Efficient Model Evaluation. *International Conference on Machine Learning*, 2021.
- Simon Lacoste-Julien, Ferenc Huszár, and Zoubin Ghahramani. Approximate inference for the loss-calibrated Bayesian. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 416–424. JMLR Workshop and Conference Proceedings, June 2011.
- Sang-woo Lee, Jin-hwa Kim, Jaehyun Jun, Jung-woo Ha, and Byoung-tak Zhang. Overcoming Catastrophic Forgetting by Incremental Moment Matching. *Advances in Neural Information Processing Systems*, 2017.
- R. Sherman Lehman. On Confirmation and Rational Betting. *The Journal of Symbolic Logic*, 20(3):251–262, 1955. ISSN 0022-4812. doi: 10.2307/2268221.
- Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports*, 7(1), December 2017.
- Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, January 1993. ISSN 08936080. doi: 10.1016/S0893-6080(05)80131-5.
- David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR

- '94, pages 3–12, Berlin, Heidelberg, August 1994. Springer-Verlag. ISBN 978-0-387-19889-7.
- Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's thesis, University of Helsinki, 1970.
- Vincenzo Lomonaco, Lorenzo Pellegrini, Pau Rodriguez, Massimo Caccia, Qi She, Yu Chen, Quentin Jodelet, Ruiping Wang, Zheda Mai, David Vazquez, German I. Parisi, Nikhil Churamani, Marc Pickett, Issam Laradji, and Davide Maltoni. CVPR 2020 Continual Learning in Computer Vision Competition: Approaches, Results, Current Challenges and Future Directions. *arXiv:2009.09929 [cs, stat]*, September 2020.
- David Lopez-paz and Marc'Aurelio Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. *Proceedings of the 31st Conference on Neural Information Processing Systems*, 31, 2017.
- Sanae Lotfi, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson. Bayesian Model Selection, the Marginal Likelihood, and Generalization, June 2022.
- Christos Louizos and Max Welling. Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors. *International Conference on Machine Learning*, pages 1708–1716, 2016.
- Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *International Conference on Machine Learning*, pages 2218–2227, July 2017.
- David MacKay. Bayesian Interpolation. *Neural Computation*, 4:415–447, 1992a.
- David Mackay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.
- David J C MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4(4):590–604, 1992b.
- David J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 1992c.
- Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A Simple Baseline for Bayesian Uncertainty in Deep Learning. *Neural Information Processing Systems*, December 2019.
- George Marsaglia. Choosing a Point from the Surface of a Sphere. *The Annals of Mathematical Statistics*, 43(2):645–646, April 1972. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177692644.

- Thomas P Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, 2001.
- Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. SLANG: Fast Structured Covariance Approximations for Bayesian Deep Learning with Natural Gradient. *arXiv:1811.04504 [cs, stat]*, January 2019.
- Andriy Mnih and Karol Gregor. Neural Variational Inference and Learning in Belief Networks. *International Conference on Machine Learning*, June 2014.
- Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the Number of Linear Regions of Deep Neural Networks. *Neural Information Processing Systems*, June 2014.
- Angel Muleshkov and Tan Nguyen. Easy Proof of the Jacobian for the N-Dimensional Polar Coordinates. *Pi Mu Epsilon Journal*, 14:269–273, 2016.
- Mervin E. Muller. A Note on a Method for Generating Points Uniformly on N-dimensional Spheres. *Commun. ACM*, 2(4):19–20, April 1959. ISSN 0001-0782. doi: 10.1145/377939.377946.
- Zachary Nado, Neil Band, Mark Collier, Josip Djolonga, Michael W. Dusenberry, Sebastian Farquhar, Angelos Filos, Marton Havasi, Rodolphe Jenatton, Ghassen Jerfel, Jeremiah Liu, Zelda Mariet, Jeremy Nixon, Shreyas Padhy, Jie Ren, Tim G. J. Rudner, Yeming Wen, Florian Wenzel, Kevin Murphy, D. Sculley, Balaji Lakshminarayanan, Jasper Snoek, Yarin Gal, and Dustin Tran. Uncertainty Baselines: Benchmarks for Uncertainty & Robustness in Deep Learning. *arXiv*, June 2021.
- Radford Neal. Bayesian Learning via Stochastic Dynamics. *Advances in Neural Information Processing Systems*, 1993.
- Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational Continual Learning. *International Conference on Learning Representations*, 2018.
- ChangYong Oh, Efstratios Gavves, and Max Welling. BOCK : Bayesian Optimization with Cylindrical Kernels. *Proceedings of The 35th International Conference on Machine Learning*, pages 3868–3877, June 2018.
- Changyong Oh, Kamil Adamczewski, and Mijung Park. Radial and Directional Posteriors for Bayesian Neural Networks. *Bayesian Deep Learning Workshop at NeurIPS*, February 2019.
- Manfred Opper. A Bayesian Approach to Online Learning. In *On-Line Learning in Neural Networks*. Cambridge University Press, 1999.

- Kazuki Osawa, Siddharth Swaroop, Anirudh Jain, Runa Eschenhagen, Richard E. Turner, Rio Yokota, and Mohammad Emtiyaz Khan. Practical Deep Learning with Bayesian Principles. *Neural Information Processing Systems*, June 2019.
- Ian Osband, John Aslanides, and Albin Cassirer. Randomized Prior Functions for Deep Reinforcement Learning. *arXiv:1806.03335 [cs, stat]*, June 2018.
- Tom Rainforth. *Automating Inference, Learning, and Design Using Probabilistic Programming*. PhD thesis, University of Oxford, 2017.
- F P Ramsey. Truth and Probability. In *Studies in Subjective Probability*. Robert E. Krieger Publishing Co, Huntington, NY, 1926.
- Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. *International Conference on Learning Representations*, February 2018.
- Larry Rendell, Raj Seshu, and David Tcheng. Layered Concept-Learning and Dynamically Variable Bias Management. *IJCAI*, pages 208–314, 1987.
- Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *International Conference on Machine Learning*, 2015.
- Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *Proceedings of The 31st International Conference on Machine Learning*, 2014.
- Hippolyt Ritter, Aleksandar Botev, and David Barber. A Scalable Laplace Approximation for Neural networks. page 15, 2018.
- Anthony Robins. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. ISSN 13600494.
- F Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.
- A. M. Saxe, J. L. McClelland, and S Ganguli. Exact solutions to the non-linear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations*, 2014.
- Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A scalable framework for continual learning. *arXiv*, 2018.

- Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*, February 2018.
- Burr Settles. Active Learning Literature Survey. *Machine Learning*, 15(2): 201–221, 2010. ISSN 00483931. doi: 10.1.1.167.4245.
- Adi Shamir, Itay Safran, Eyal Ronen, and Orr Dunkelman. A Simple Explanation for the Existence of Adversarial Examples with Small Hamming Distance. *arXiv*, January 2019.
- Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep Active Learning for Named Entity Recognition. In *International Conference on Learning Representations*, 2018.
- Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. *Advances In Neural Information Processing Systems*, pages 2994–3003, 2017.
- Aditya Siddhant and Zachary C. Lipton. Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1318.
- Samrath Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational Adversarial Active Learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5971–5980, Seoul, Korea (South), October 2019. IEEE. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.00607.
- Brian Skyrms. Coherence. In *Scientific Inquiry in Philosophical Perspective*. University of Pittsburgh Press, Pittsburgh, 1987.
- M. D. Springer and W. E. Thompson. The Distribution of Products of Beta, Gamma and Gaussian Random Variables. *SIAM Journal on Applied Mathematics*, 18(4):721–737, 1970. ISSN 0036-1399.
- Masashi Sugiyama and Cs Titech Ac Jp. Active Learning in Approximately Linear Regression Based on Conditional Expectation of Generalization Error. *Journal of Machine Learning Research*, 7:141–166, 2006.
- Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning Structured Weight Uncertainty in Bayesian Neural Networks. *Artificial Intelligence and Statistics*, pages 1283–1292, 2017.
- Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational Bayesian Neural Networks. *International Conference on Learning Representations*, 2019.

- Jakub Swiatkowski, Kevin Roth, Bastiaan S. Veeling, Linh Tran, Joshua V. Dillon, Jasper Snoek, Stephan Mandt, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. The k-tied Normal Distribution: A Compact Parameterization of Gaussian Mean Field Posteriors in Bayesian Neural Networks. *International Conference on Machine Learning*, 2020.
- Naftali Tishby, Esther Levin, and Sara A. Solla. Consistent inference of probabilities in layered networks. *International Joint Conf Neural Network*, pages 403–409, December 1989.
- Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-Effective Active Learning for Deep Image Classification. *IEEE Transactions on Circuits and Systems for Video Technology*, January 2017.
- Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. *Proceedings of the 28th International Conference on Machine Learning*, 2011.
- Si Wen, Tahsin M. Kurc, Le Hou, Joel H. Saltz, Rajarsi R. Gupta, Rebecca Batiste, Tianhao Zhao, Vu Nguyen, Dimitris Samaras, and Wei Zhu. Comparison of Different Classifiers with Active Learning to Support Quality Control in Nucleus Segmentation in Pathology Images. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2017:227–236, 2018a. ISSN 2153-4063.
- Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. *International Conference on Learning Representations*, April 2018b.
- Gregory Wheeler. Bounded Rationality. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2020 edition, 2020.
- Andrew Gordon Wilson and Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. *International Conference on Machine Learning*, April 2020.
- Andrew Gordon Wilson, Pavel Izmailov, Matthew D Hoffman, Yarin Gal, Yingzhen Li, Melanie F Pradier, Sharad Vikram, Andrew Foong, Sanae Lotfi, and Sebastian Farquhar. Evaluating Approximate Inference in Bayesian Deep Learning. *NeurIPS Competition*, 2021.
- Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, Jose Miguel Hernandez-Lobato, and Alexander L Gaunt. Deterministic Variational Inference for Robust Bayesian Neural Networks. *International Conference on Learning Representations*, 2019.

- Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv*, August 2017.
- Jie Yang, Thomas Drake, Andreas Damianou, and Yoelle Maarek. Leveraging Crowdsourcing Data for Deep Active Learning An Application: Learning Intents in Alexa. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, pages 23–32, Republic and Canton of Geneva, CHE, April 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5639-8. doi: 10.1145/3178876.3186033.
- Donggeun Yoo and In So Kweon. Learning Loss for Active Learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 93–102, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.00018.
- Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. *Proceedings of the 34th International Conference on Machine Learning*, pages 3987–3995, 2017.
- Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy Natural Gradient as Variational Inference. *International Conference on Machine Learning*, 2018.
- Yao Zhang and Alpha A. Lee. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *Chemical Science*, 10(35):8154–8163, September 2019. ISSN 2041-6539. doi: 10.1039/C9SC00616H.
- Yichuan Zhang and Charles Sutton. Semi-Separable Hamiltonian Monte Carlo for Inference in Bayesian Hierarchical Models. *Neural Information Processing Systems*, June 2014.

Appendix A

Appendix to Chapter 3

A.1 Experimental Details

Full Description of Covariance Visualization

Here we provide details on the method used to produce fig. 3.5. The linear version of the visualization is discussed in section 3.3 and the piecewise-linear version is discussed in section 3.3.

In all cases, we train a neural network using mean-field variational inference in order to visualize the covariance of the product matrix. The details of training are provided in table A.2. The product matrix is calculated from the weight matrices of an L -layer network. In the linear case, this is just the matrix product of the L layers. In the piecewise-linear case the definition of the product matrix is described in more detail in appendix A.2. All covariances are calculated using 10,000 samples from the converged approximate posterior. Note that for L weight matrices there are $L - 1$ layers of hidden units.

We compare these learned product matrices, in fig. 3.5f, to a randomly sampled product matrix. To do so, we sample weight layers whose entries are distributed normally. Each weight is sampled with standard deviation 0.3 and with a mean 0.01 and each weight matrix is 16x16. This visualization is with a linear product matrix of 5 layers.

Effect of Depth Measured on Iris Experimental Settings

We describe the full- and diagonal-covariance experiment settings in table A.3. We use a very small model on a small dataset because full-covariance variational inference is unstable, requiring a matrix inversion of a K^4 matrix for hidden unit width K . Unfortunately, for deeper models the initializations still

Hyperparameter	Setting description
Architecture	MLP
Number of hidden layers	3
Layer Width	100
Activation	Leaky ReLU
Approximate Inference Algorithm	Mean-field VI (Flipout [Wen et al., 2018b])
Optimization algorithm	Amsgrad [Reddi et al., 2018]
Learning rate	10^{-3}
Batch size	250
Variational training samples	1
Variational test samples	1
Temperature	65
Noise scale	0.05
Epochs	6000
Variational Posterior Initialization	Tensorflow Probability default
Prior	$\mathcal{N}(0, 1.0^2)$
Dataset	Toy (see text)
Number of training runs	1
Number of evaluation runs	1
Measures of central tendency	n.a.
Runtime per result	< 5m
Computing Infrastructure	Nvidia GTX 1060

Table A.1: Experimental Setting—Toy Regression Visualization. Note that for these visualizations we are purely demonstrating the possibility of in-between uncertainty. As a result, a single training/evaluation run suffices to make an existence claim, so we do not do multiple runs in order to calculate a measure of central tendency.

resulted in failed training for some seeds. To avoid this issue, we selected the 10 best seeds out of 100 training runs, and report the mean and standard error for these. Because we treat full- and diagonal-covariance in the same way, the resulting graph is a fair reflection of their relative best-case merits, but not intended as anything resembling a ‘real-world’ performance benchmark.

Readers may consider the Iris dataset to be unhelpfully small, however this was a necessary choice. We note that the small number of training points creates a broad posterior, which is the best-case scenario for a full-covariance approximate posterior.

HMC Experimental Settings

We begin by sampling from the true posterior using HMC.

We use the simple two-dimensional binary classification ‘make moons’ task.

Hyperparameter	Setting description
Architecture	MLP
Number of hidden layers	0-9
Layer Width	16
Activation	Linear or Leaky Relu with $\alpha = 0.1$
Approximate Inference Algorithm	Mean-field Variational Inference
Optimization algorithm	Amsgrad [Reddi et al., 2018]
Learning rate	10^{-3}
Batch size	64
Variational training samples	16
Variational test samples	16
Epochs	10
Variational Posterior Initial Mean	He et al. [2016]
Variational Posterior Initial Standard Deviation	$\log[1 + e^{-3}]$
Prior	$\mathcal{N}(0, 0.23^2)$
Dataset	FashionMNIST [Xiao et al., 2017]
Preprocessing	Data normalized $\mu = 0, \sigma = 1$
Validation Split	90% train - 10% validation
Number of training runs	1
Number of evaluation runs	1
Measures of central tendency	n.a.
Runtime per result	< 3m
Computing Infrastructure	Nvidia GTX 1080

Table A.2: Experimental Setting—Covariance Visualization. Note that for these visualizations we are purely demonstrating the possibility of off-diagonal covariance. As a result, a single training/evaluation run suffices to make an existence claim, so we do not do multiple runs in order to calculate a measure of central tendency.

We use 500 training points (generated using `random_state = 0`).¹ Using Cobb et al. [2019b], we apply the No-U-turn Sampling scheme [Hoffman and Gelman, 2014] with an initial step size of 0.01. We use a burn-in phase with 10,000 steps targetting a rejection rate of 0.8. We then sample until we collect 1,000 samples from the true posterior, taking 100 leapfrog steps in between every sample used in order to ensure samples are less correlated. For each result, we recalculate the HMC samples 20 times with a different random seed. All chains are initialized at the optimum of a mean-field variational inference model in order to help HMC rapidly find a mode of the true posterior. We use a prior precision, normalizing constant, and τ of 1.0. The model is designed to have as close to 1000 non-bias parameters each time as possible, adjusting the width given the depth of the model. We observe that the accuracies for

¹https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html#sklearn.datasets.make_moons

Hyperparameter	Setting description
Architecture	MLP
Number of hidden layers	1-4
Layer Width	4
Activation	Leaky Relu
Approximate Inference Algorithm	Variational Inference
Optimization algorithm	Amsgrad [Reddi et al., 2018]
Learning rate	10^{-3}
Batch size	16
Variational training samples	1
Variational test samples	1
Epochs	1000 (early stopping patience=30)
Variational Posterior Initial Mean	He et al. [2016]
Variational Posterior Initial Standard Deviation	$\log[1 + e^{-6}]$
Prior	$\mathcal{N}(0, 1.0^2)$
Dataset	Iris [Xiao et al., 2017]
Preprocessing	None.
Validation Split	100 train - 50 test
Number of training runs	100
Number of evaluation runs	100
Measures of central tendency	(See text.)
Runtime per result	< 5m
Computing Infrastructure	Nvidia GTX 1080

Table A.3: Experimental Setting—Full Covariance.

# Hidden Layers	ReLU		Leaky ReLU 0.5		Leaky ReLU 0.95		Linear	
	Test Acc.	Acceptance	Test Acc.	Acceptance	Test Acc.	Acceptance	Test Acc.	Acceptance
1	99.1%	84.8%	98.4%	84.9%	91.9%	85.4%	83.9%	78.0%
2	99.7%	77.0%	99.5%	73.9%	96.4%	76.3%	84.2%	44.4%
3	99.1%	58.0%	99.6%	46.3%	97.2%	74.5%	84.4%	37.0%
4	99.5%	62.2%	99.6%	50.9%	95.8%	68.2%	84.4%	43.2%
5	98.1%	61.8%	99.5%	53.8%	98.4%	62.4%	84.3%	35.2%
6	95.4%	78.5%	99.6%	51.0%	98.0%	62.6%	84.1%	33.7%
7	92.7%	68.1%	99.7%	54.6%	97.5%	59.7%	84.0%	33.0%
8	87.8%	68.3%	99.6%	49.7%	98.0%	62.5%	83.8%	36.4%
9	80.6%	73.9%	99.6%	46.3%	97.4%	60.2%	83.9%	36.5%
10	74.6%	74.9%	99.5%	45.7%	97.1%	61.8%	83.8%	40.4%

Table A.4: HMC samples for ReLU networks are most accurate for smaller numbers of layers, the samples from deeper models may therefore be slightly less reliable. Acceptance rates tend to be with 10-20 percentage points of 65%, regarded as a good balance of exploration to avoiding unnecessary resampling. The more linear models are less accurate, as one would expect for a dataset that is not linearly separable.

the ReLU network fail for the deeper models, suggesting that after about 7 layers the posterior estimate may become slightly less reliable (see table A.4). Acceptance rates are broadly in a sensible region for most of the chains.

Using these samples, we find a Gaussian fit. For each model we fit a Gaussian mixture model with between 1 and 4 components and pick the one with the best Bayesian information criterion (see fig. 3.1). We then find the best diagonal fit to this distribution, which is a Gaussian distribution with the same mean and with a precision matrix equal to the inverse of the diagonal precision of the full-covariance Gaussian. We do this because variational inference uses the mode-seeking KL-divergence, so we are interested in the properties of a single Gaussian mode. The overall empirical covariance would lead to a mode-covering distribution, while optimizing the mode-seeking KL to the empirical distribution would of course result in a point-sized distribution centred at one of the HMC samples. Using one mode of a mixture of Gaussians is therefore the closest we can come to finding a single mode of the true posterior of the sort that VI might uncover.

Finally, we calculate the KL divergence between these two distributions. The graph reports the mean and shading reflects the standard error of the mean, though note that because all runs are initialized from the same point, this underestimates the overall standard error. All experiments in this and other sections were run on a desktop workstation with an Nvidia 1080 GPU.

For the Wasserstein divergence, we estimate the distance between the empirical distributions formed by the HMC samples and samples from the full- and diagonal-covariance posterior approximation. We used the Python Optimal Transport package to estimate the divergence.

Diagonal- and Structured-SWAG at Varying Depths

We use the implementation of SWAG available publicly at https://github.com/wjmaddox/swa_gaussian. We adapt their code to vary the depth of the PreResNet architecture for the values 2, 8, 14, 20, 26, 32, 38. We use the hyperparameter settings used by Maddox et al. [2019] for PreResNet154 on CIFAR100 on all datasets to train the models. We use 10 seeds to generate the error bars, which are plotted with one standard deviation. We use the same SWAG run to fit both the full and diagonal approximations, and use 30 samples in the forward pass.

A.2 Proofs

Full Derivation of the Product Matrix Covariance

Proposition 2. *For $L \geq 3$, the product matrix $M^{(L)}$ of factorized weight matrices can have non-zero covariance between any and all pairs of elements. That is, there exists a set of mean-field weight matrices $\{W^{(l)} | 1 \leq l < L\}$ such*

that $M^{(L)} = \prod W^{(l)}$ and the covariance between any possible pair of elements of the product matrix:

$$\text{Cov}(m_{ab}^{(L)}, m_{cd}^{(L)}) \neq 0, \quad (3.5)$$

where $m_{ij}^{(L)}$ are elements of the product matrix in the i^{th} row and j^{th} column, and for any possible indexes $a, b, c,$ and d .

Proof. We begin by explicitly deriving the covariance between elements of the product matrix.

Consider the product matrix, $M^{(L)}$, which is the matrix product of an arbitrary weight matrix, $W^{(L)}$, with a mean field distribution over it's entries, and the product matrix with one fewer layers, $M^{(L-1)}$. Expressed in terms of the elements of each matrix in row-column notation this matrix multiplication can be written:

$$m_{ab}^{(L)} = \sum_{i=1}^{K_{L-1}} w_{ai}^{(L)} m_{ib}^{(L-1)}. \quad (A.1)$$

We make no assumption about K_{L-1} except that it is non-zero and hence the weights can be any rectangular matrix.² The weight matrix $W^{(L)}$ is assumed to have a mean-field distribution (the covariance matrix is zero for all off diagonal elements) with arbitrary means:

$$\begin{aligned} \text{Cov}(w_{ac}^{(L)}, w_{bd}^{(L)}) &= \Sigma_{abcd}^{(L)} = \delta_{ac}\delta_{bd}\sigma_{ab}^{(L)}; \\ \mathbb{E} [w^{(L)}]_{ab} &= \mu_{ab}^{(L)}. \end{aligned} \quad (A.2)$$

δ are the Kronecker delta. Note that the weight matrix is 2-dimensional, but the covariance matrix is defined between every element of the weight matrix. While it can be helpful to regard it as 2-dimensional also, we index it with the four indices that define a pair of elements of the weight matrix.

We begin by deriving the expression for the covariance of the L -layer product matrix $\text{Cov}(m_{ab}^{(L)}, m_{cd}^{(L)})$. Using the definition of the product matrix in eq. (A.1):

$$\text{Cov}(m_{ab}^{(L)}, m_{cd}^{(L)}) = \text{Cov}\left(\sum_i w_{ai}^{(L)} m_{ib}^{(L-1)}, \sum_j w_{cj}^{(L)} m_{jd}^{(L-1)}\right). \quad (A.3)$$

We then simplify this using the linearity of covariance (for brevity call the covariance of the product matrix $\hat{\Sigma}_{abcd}^{(L)}$):

$$\hat{\Sigma}_{abcd}^{(L)} = \sum_{ij} \text{Cov}(w_{ai}^{(L)} m_{ib}^{(L-1)}, w_{cj}^{(L)} m_{jd}^{(L-1)}), \quad (A.4)$$

²We set aside bias parameters, as they complicate the algebra, but adding them only strengthens the result because each bias term affects an entire row.

rewriting using the definition of covariance in terms of a difference of expectations:

$$\begin{aligned}
&= \sum_{ij} \mathbb{E} \left[[w_{ai}^{(L)} m_{ib}^{(L-1)} w_{cj}^{(L)} m_{jd}^{(L-1)}] \right] \\
&\quad - \mathbb{E} \left[[w_{ai}^{(L)} m_{ib}^{(L-1)}] \right] \mathbb{E} \left[[w_{cj}^{(L)} m_{jd}^{(L-1)}] \right], \tag{A.5}
\end{aligned}$$

using the fact that by assumption the new layer is independent of the previous product matrix:

$$\begin{aligned}
&= \sum_{ij} \mathbb{E} \left[[w_{ai}^{(L)} w_{cj}^{(L)}] \right] \mathbb{E} \left[[m_{ib}^{(L-1)} m_{jd}^{(L-1)}] \right] \\
&\quad - \mathbb{E} \left[[w_{ai}^{(L)}] \right] \mathbb{E} \left[[w_{cj}^{(L)}] \right] \mathbb{E} \left[[m_{ib}^{(L-1)}] \right] \mathbb{E} \left[[m_{jd}^{(L-1)}] \right], \tag{A.6}
\end{aligned}$$

and rewriting to expose the dependence on the covariance of $M^{(L-1)}$:

$$\begin{aligned}
&= \sum_{ij} \left(\mathbb{E} \left[[w_{ai}^{(L)} w_{cj}^{(L)}] \right] - \mathbb{E} \left[[w_{ai}^{(L)}] \right] \mathbb{E} \left[[w_{cj}^{(L)}] \right] \right) \\
&\quad \cdot \left(\mathbb{E} \left[[m_{ib}^{(L-1)} m_{jd}^{(L-1)}] \right] - \mathbb{E} \left[[m_{ib}^{(L-1)}] \right] \mathbb{E} \left[[m_{jd}^{(L-1)}] \right] \right) \\
&\quad + \mathbb{E} \left[[w_{ai}^{(L)}] \right] \mathbb{E} \left[[w_{cj}^{(L)}] \right] \left(\mathbb{E} \left[[m_{ib}^{(L-1)} m_{jd}^{(L-1)}] \right] - \mathbb{E} \left[[m_{ib}^{(L-1)}] \right] \mathbb{E} \left[[m_{jd}^{(L-1)}] \right] \right) \\
&\quad + \mathbb{E} \left[[m_{ib}^{(L-1)}] \right] \mathbb{E} \left[[m_{jd}^{(L-1)}] \right] \left(\mathbb{E} \left[[w_{ai}^{(L)} w_{cj}^{(L)}] \right] - \mathbb{E} \left[[w_{ai}^{(L)}] \right] \mathbb{E} \left[[w_{cj}^{(L)}] \right] \right), \tag{A.7}
\end{aligned}$$

substituting the covariance:

$$\begin{aligned}
&= \sum_{ij} \text{Cov} \left(w_{ai}^{(L)}, w_{cj}^{(L)} \right) \cdot \text{Cov} \left(m_{ib}^{(L-1)}, m_{jd}^{(L-1)} \right) \\
&\quad + \mathbb{E} \left[[w_{ai}^{(L)}] \right] \mathbb{E} \left[[w_{cj}^{(L)}] \right] \text{Cov} \left(m_{ib}^{(L-1)}, m_{jd}^{(L-1)} \right) \\
&\quad + \mathbb{E} \left[[m_{ib}^{(L-1)}] \right] \mathbb{E} \left[[m_{jd}^{(L-1)}] \right] \text{Cov} \left(w_{ai}^{(L)}, w_{cj}^{(L)} \right). \tag{A.8}
\end{aligned}$$

This gives us a recursive expression for the covariance of the product matrix.

It is straightforward to substitute in our expressions for mean and variance in a mean-field network provided in eq. (A.2), where we use the fact that the initial $M^{(1)}$ product matrix is just a single mean-field layer.

In this way, we show that:

$$\begin{aligned}
\hat{\Sigma}_{abcd}^{(2)} &= \sum_{ij} \left(\delta_{ac} \delta_{ij} \sigma_{ai}^{(2)} \right) \cdot \left(\delta_{ij} \delta_{bd} \sigma_{ib}^{(1)} \right) \\
&\quad + \mu_{ai}^{(2)} \mu_{cj}^{(2)} \left(\delta_{ij} \delta_{bd} \sigma_{ib}^{(1)} \right) + \mathbb{E} \left[[m_{ib}^{(1)}] \right] \mathbb{E} \left[[m_{jd}^{(1)}] \right] \left(\delta_{ac} \delta_{ij} \sigma_{ai}^{(2)} \right) \tag{A.9}
\end{aligned}$$

$$= \sum_i \delta_{ac} \delta_{bd} \sigma_{ai}^{(2)} \sigma_{ib}^{(1)} + \delta_{bd} \mu_{ai}^{(2)} \mu_{ci}^{(2)} \sigma_{ib}^{(1)} + \delta_{ac} \mu_{ib}^{(1)} \mu_{id}^{(1)} \sigma_{ai}^{(2)}. \tag{A.10}$$

The first term of eq. (A.10) has the Kronecker deltas $\delta_{ac}\delta_{bd}$ meaning that it contains diagonal entries in the covariance matrix. The second term has only δ_{bd} meaning it contains entries for the covariance between weights that share a column. The third term has only δ_{ac} meaning it contains entries for the covariance between weights that share a row.

This covariance of the product matrix already has some off-diagonal terms, but it does not yet contain non-zero covariance for weights that share neither a row nor a column.

But we can repeat the process and find $\hat{\Sigma}_{abcd}^{(3)}$ using eq. (A.8) and our expression for $\hat{\Sigma}_{ibjd}^{(2)}$:

$$\begin{aligned} \hat{\Sigma}_{abcd}^{(3)} &= \sum_{ij} \left(\delta_{ac}\delta_{ij}\sigma_{ai}^{(3)} \right) \cdot \hat{\Sigma}_{ibjd}^{(2)} + \mu_{ai}^{(3)}\mu_{cj}^{(3)}\hat{\Sigma}_{ibjd}^{(2)} \\ &\quad + \mathbb{E} \left[[m_{ib}^{(2)}] \right] \mathbb{E} \left[[m_{jd}^{(2)}] \right] \left(\delta_{ac}\delta_{ij}\sigma_{ai}^{(3)} \right) \end{aligned} \quad (\text{A.11})$$

$$\begin{aligned} &= \sum_{ij} \left(\delta_{ac}\delta_{ij}\sigma_{ai}^{(3)} \right) \cdot \sum_k \left(\delta_{ij}\delta_{bd}\sigma_{ik}^{(2)}\sigma_{kb}^{(1)} + \delta_{bd}\mu_{ik}^{(2)}\mu_{jk}^{(2)}\sigma_{kb}^{(1)} + \delta_{ij}\mu_{kb}^{(1)}\mu_{kd}^{(1)}\sigma_{ik}^{(2)} \right) \\ &\quad + \mu_{ai}^{(3)}\mu_{cj}^{(3)} \cdot \sum_k \left(\delta_{ij}\delta_{bd}\sigma_{ik}^{(2)}\sigma_{kb}^{(1)} + \delta_{bd}\mu_{ik}^{(2)}\mu_{jk}^{(2)}\sigma_{kb}^{(1)} + \delta_{ij}\mu_{kb}^{(1)}\mu_{kd}^{(1)}\sigma_{ik}^{(2)} \right) \\ &\quad + \mu_{ib}^{(2)}\mu_{jd}^{(2)} \left(\delta_{ac}\delta_{ij}\sigma_{ai}^{(3)} \right). \end{aligned} \quad (\text{A.12})$$

It is the term in red which has no factors of Kronecker deltas in any of the indices a, b, c, or d. It is therefore present in all elements of the covariance matrix of the product matrix, regardless of whether they share one or both index. This shows that, so long as the distributional parameters themselves are non-zero, the product matrix can have a fully non-zero covariance matrix for $L = 3$.

We note that there are many weight matrices for which the resulting covariance is non-zero everywhere—we think this is actually typical. Indeed, empirically, we found that for any network we cared to construct, we were unable to find covariances that *were* zero anywhere. However, for our existence proof, we simply note that for any matrix in which all the means are positive each term of the resulting expression is positive (the standard deviation parameters may be taken as positive without loss of generality). In that case, it is impossible that any term cancels with any other, so the resulting covariance is positive everywhere.

Last, we examine the recurrence relationship in eq. (A.8). Once it is the case that $\text{Cov}(m_{ib}^{(L-1)}, m_{jd}^{(L-1)}) \neq 0$ for all possible indices, the covariances between elements of $M^{(L)}$ may also be non-zero. Observe simply that if the means of the top weight matrix are positive, then each of the terms in eq. (A.8) are positive, so it is impossible for any term to cancel out with any other. The fact that the elements of $M^{(L-1)}$ have non-zero covariances everywhere

therefore entails that there is a weight matrix $W^{(L)}$ such that $M^{(L)}$ has non-zero covariance between all of its elements also, as required.

Remark 2. *Here, we show only an existence proof, and therefore we restrict ourselves to positive means and standard deviations to simplify the proof. In fact, we believe that non-zero covariance is the norm, rather than a special case, and found this in all our numerical simulations for both trained and randomly sampled models. However, we do not believe that (in the linear case) any covariance matrix can be created from a deep mean-field product.*

□

Matrix Variate Gaussian as a Special Case of Three-Layer Product Matrix

We can gain insight into the richness of the possible covariances by considering the limited case of the product matrix $M^{(3)} = ABC$ where B is a matrix whose elements are independent Gaussian random variables and A and C are deterministic. We note that this is a highly constrained setting, and that the covariances which can be induced with A and C as random variables have the more complex form shown in appendix A.2. We can show the following:

Proposition 3. *The Matrix Variate Gaussian (Kronecker-factored) distribution is a special case of the distribution over elements of the product matrix. In particular, for $M^{(3)} = ABC$, $M^{(3)}$ is distributed as an MVG random variable when A and C are deterministic and B has its elements distributed as fully factorized Gaussians with unit variance.*

Proof. Consider the product matrix $M^{(3)} = ABC$. where B is a matrix whose elements are independent Gaussian random variables and A and C are deterministic. The elements of B are distributed with mean μ_B and have a diagonal covariance matrix Σ_B .

We begin by recalling the property of the Kronecker product that:

$$\text{vec}(ABC) = (C^\top \otimes A)\text{vec}(B). \quad (\text{A.13})$$

By definition $\text{vec}(M^{(3)}) = \text{vec}(ABC) = (C^\top \otimes A)\text{vec}(B)$. Because $C^\top \otimes A$ is deterministic, it follows from a basic property of the covariance that the covariance of the product matrix $\Sigma_{M^{(3)}}$ is given by:

$$\Sigma_{M^{(3)}} = (C^\top \otimes A)\Sigma_B(C^\top \otimes A)^\top. \quad (\text{A.14})$$

Using the fact that the transpose is distributive over the Kronecker product, this is equivalent to:

$$\Sigma_{M^{(3)}} = (C^\top \otimes A)\Sigma_B(C \otimes A^\top). \quad (\text{A.15})$$

Because we only want to establish that the family of distributions expressible contains the matrix variate Gaussians, we do not need to use all the possible freedom, and we can set $\Sigma_B = I$. In this special case:

$$\Sigma_{M^{(3)}} = (C^\top \otimes A)(C \otimes A^\top). \quad (\text{A.16})$$

Using the mixed-product property, this is equivalent to:

$$\Sigma_{M^{(3)}} = (C^\top C) \otimes (AA^\top). \quad (\text{A.17})$$

Now, we note that any positive semi-definite matrix can be written in the form $A = M^\top M$, so this implies that, defining the positive semi-definite matrices $V = C^\top C$ and $U = AA^\top$, we have that the covariance $\Sigma_{M^{(3)}}$ is of the form,

$$\Sigma_{M^{(3)}} = V \otimes U. \quad (\text{A.18})$$

Similarly, we can consider the mean of the product matrix $\mu_{M^{(3)}}$. From eq. (A.13), we can see that:

$$\mu_{M^{(3)}} = (C^\top \otimes A)\text{vec}(\mu_B). \quad (\text{A.19})$$

But since we have not yet constrained μ_B , it is clear that this allows us to set any $\mu_{M^{(3)}}$ we desire by choosing $\mu_B = (C^\top \otimes A)^{-1}\mu_{M^{(3)}}$.

So far, we have only discussed the first- and second-moments, and the proof has made no assumptions about specific distributions. However, we now observe that a random variable X is distributed according to the Matrix Variate Gaussian distribution according to some mean μ_X and with scale matrices U and V if and only if $\text{vec}(X)$ is a multivariate Gaussian with mean $\vec{\mu}_X$ and covariance $U \otimes V$.

Therefore, given eq. (A.19) and eq. (A.18), the special case of $M^{(3)}$ where the first and last matrices are deterministic and the middle layer has a fully-factorized Gaussian distribution over the weights with unit variance is a Matrix Variate Gaussian distribution where:

$$\text{vec}(\mu_X) = (C^\top \otimes A)\text{vec}(\mu_B); \quad (\text{A.20})$$

$$V = C^\top C; \quad (\text{A.21})$$

$$U = A^\top A. \quad (\text{A.22})$$

□

Proof of Linearized Product Matrix Covariance

Proof of Local Linearity

We consider local linearity in the case of piecewise-linear activations like ReLU.

Lemma 1. Consider an input point $\mathbf{x}^* \in \mathcal{D}$. Consider a realization of the model weights θ . Then, for any \mathbf{x}^* , the neural network function f_θ is linear over some compact set $\mathcal{A}_\theta \subset \mathcal{D}$ containing \mathbf{x}^* . Moreover, \mathcal{A}_θ has non-zero measure for almost all \mathbf{x}^* w.r.t. the Lebesgue measure.

Proof. Neural networks with finitely many piecewise-linear activations are themselves piecewise-linear. Therefore, for a finite neural network, we can decompose the input domain \mathcal{D} into regions $\mathcal{D}_i \subseteq \mathcal{D}$ such that

1. $\cup \mathcal{D}_i = \mathcal{D}$,
2. $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset \quad \forall i \neq j$,
3. f_θ is a linear function on points in \mathcal{D}_i for each i .

For a finite neural network, there are at most finitely many regions \mathcal{D}_i . In particular, with hidden layer widths n_i in the i 'th layer, with an input domain \mathcal{D} with dimension n_0 , Montúfar et al. [2014] show that the network can define maximally a number of regions in input space bounded above by:

$$\left(\prod_{i=1}^{L-1} \left\lfloor \frac{n_i}{n_0} \right\rfloor^{n_0} \right) \sum_{j=0}^{n_0} \binom{n_L}{j}. \quad (\text{A.23})$$

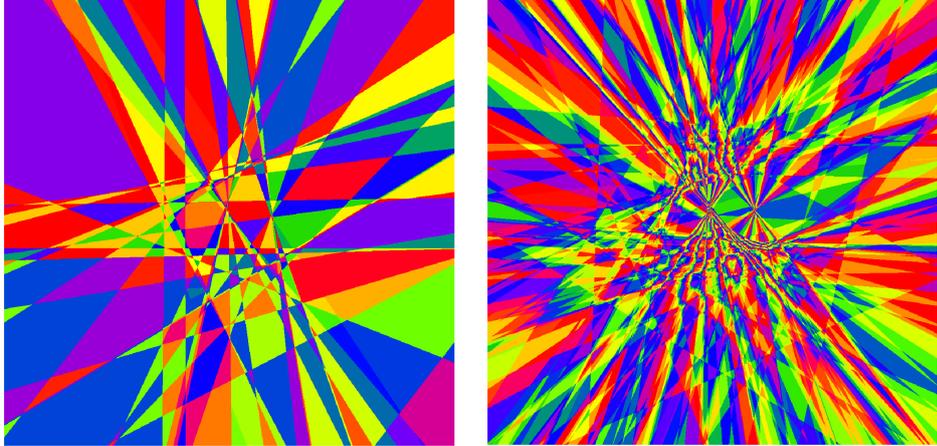
Except in the trivial case where the input domain has measure zero, this along with (1) and (2) jointly entail that at least one of the regions \mathcal{D}_i has non-zero measure. This, with (3) entails that only a set of input points of zero measure do *not* fall in a linear region of non-zero measure. These points correspond to inputs that lie directly on the inflection points of the ReLU activations. \square

We visualize \mathcal{A}_{θ_i} in fig. A.1a. This shows a two-dimensional input space (from the two moons dataset). Parts of the space within which a neural network function is linear are shown in one color. The regions are typically smallest where the most detail is required in the trained function.

Defining the Local Product Matrix

We define a random variate representing the local product matrix, for an input point \mathbf{x}^* , using the following procedure.

To draw a finite N samples of the random variate, we sample N realizations of the weight parameters $\Theta = \{\theta_i \text{ for } 1 \leq i \leq N\}$. For each θ_i , given \mathbf{x}^* there is a compact set $\mathcal{A}_{\theta_i} \subset \mathcal{D}$ within which f_{θ_i} is linear (and $\mathbf{x}^* \in \mathcal{A}_i$) by lemma 1. Therefore, all samples of the neural network function are linear in the intersection region $\mathcal{A} = \bigcap_i \mathcal{A}_{\theta_i}$. We note that \mathcal{A} at least contains \mathbf{x}^* . Moreover, so long as \mathcal{D} is a compact subset of the reals, \mathcal{A} has non-zero



(a) 1 sample: \mathcal{A}_{θ_i}

(b) 5 samples: $\mathcal{A} = \bigcap_{0 \leq i < 5} \mathcal{A}_{\theta_i}$

Figure A.1: Visualizaton of the linear regions in input-space for a two-dimensional binary classification problem (two moons). Colored regions show contiguous areas within which a neural network function is linear. We use an arbitrary numerical encoding of these regions (we interpret the sign pattern of activated relus as an integer in base 2) and a cyclic colour scheme for visualisation, so the color of each region is arbitrary, and two non-contiguous regions with the same color are not the same region. The neural network has one hidden layer with 100 units and is trained for 1000 epochs on 500 datapoints from scipy’s two moons using Adam. (a) a single model has fairly large linear regions, with the most detail clustered near the region of interest. (b) The regions within which all samples are linear (the intersection set \mathcal{A}) are smaller, but finite. The local product matrix is valid within one of these regions for any input point.

measure.³ Figure A.1b shows a visualization of \mathcal{A} with 5 samples. The linear regions are smaller, because there is a discontinuity if any of the models is

³Intuitively, we know that \mathbf{x}^* is in all \mathcal{A}_{θ_i} , so when we add a new sample we know that there is either overlap around \mathbf{x}^* or the point \mathbf{x}^* is on the boundary of the new subset, which means we could equally well pick a different set that has \mathbf{x}^* on its boundary and *does* have non-zero-measure overlap with the previous sets.

More formally, consider some compact set $\mathcal{A}_{\theta_0} \subset \mathcal{D}$ with non-zero measure such that $\mathbf{x}^* \in \mathcal{A}_{\theta_0}$. Take some new compact set $\mathcal{A}_{\theta_1} \subset \mathcal{D}$ with non-zero measure also such that $\mathbf{x}^* \in \mathcal{A}_{\theta_1}$. Define the intersection between those sets $\mathcal{B} = \mathcal{A}_{\theta_0} \cap \mathcal{A}_{\theta_1}$. Suppose that \mathcal{B} has zero measure. But both \mathcal{A}_{θ_0} and \mathcal{A}_{θ_1} contain \mathbf{x}^* , so the only way that \mathcal{B} could have zero measure is if \mathbf{x}^* is an element in the boundary of both sets. But if \mathcal{A}_{θ_1} has \mathbf{x}^* on its boundary, then, by the continuity of the real space, there is at least one other compact set \mathcal{A}'_{θ_1} , different to \mathcal{A}_{θ_1} , such that \mathbf{x}^* is on its boundary. But, since by hypothesis \mathcal{A}_{θ_0} has non-zero measure, there exists such a set \mathcal{A}'_{θ_1} which has a non-zero-measure intersection with \mathcal{A}_{θ_0} . We can therefore select \mathcal{A}'_{θ_1} instead of \mathcal{A}_{θ_1} when building \mathcal{A} , such that the intersection with \mathcal{A}_{θ_0} has non-zero measure. By repeated application of this argument, we can guarantee that for any finite Θ we are able to find a set of $\mathcal{A}_{\theta_i} \subset \mathcal{D}$ such that $\forall i : \mathbf{x}^* \in \mathcal{A}_{\theta_i}$ and \mathcal{A} has non-zero measure. This argument does not guarantee that the measure of \mathcal{A} in the limit as N tends to infinity is non-zero.

discontinuous. Nevertheless, the space is composed of regions of finite size within which the neural network function is linear.

For each θ_i we can compute a local product matrix within \mathcal{A} . Ordinarily, setting aside the bias term for simplicity, a neural network hidden layer \mathbf{h}_{l+1} can be written in terms of the hidden layer before it, a weight matrix W_l , and an activation function.

$$\mathbf{h}_{l+1} = \sigma(W_l \mathbf{h}_l) \quad (\text{A.24})$$

We observe that within \mathcal{A} the activation function becomes linear. This allows us to define an activation vector $\mathbf{a}_{\mathbf{x}^*}$ within \mathcal{A} such that the equation can be written:

$$\mathbf{h}_{l+1} = \mathbf{a}_{\mathbf{x}^*} \cdot (W_l \mathbf{h}_l). \quad (\text{A.25})$$

The activation vector can be easily calculated by calculating $W_l \mathbf{h}_l$, seeing which side of the (Leaky) ReLU the activation is on within that linear region for each hidden unit, and selecting the correct scalar (0 or 1 for a ReLU, or α or 1 for a Leaky ReLU).

This allows us to straightforwardly construct a product matrix for each θ_i which takes the activation function into account (in the linear case, we effectively always set $\mathbf{a}_{\mathbf{x}^*}$ to equal the unit vector). The random variate $P_{\mathbf{x}^*}$ is constructed with these product matrices for realizations of the weight distribution.

Samples from the resulting random variate $P_{\mathbf{x}^*}$ are therefore distributed such that samples from $P_{\mathbf{x}^*} \mathbf{x}^*$ have the same distribution as samples of the predictive posterior y given \mathbf{x}^* within \mathcal{A} .

Proof that the Local Product Matrix has Non-zero Off-diagonal Covariance

Proposition 4. *Given a mean-field distribution over the weights of neural network f with piecewise linear activations, f can be written in terms of the local product matrix $P_{\mathbf{x}^*}$ within \mathcal{A} .*

For $L \geq 3$, for activation functions which are non-zero everywhere, there exists a set of weight matrices $\{W^{(l)} | 1 \leq l < L\}$ such that all elements of the local product matrix have non-zero off-diagonal covariance:

$$\text{Cov}(p_{ab}^{\mathbf{x}^*}, p_{cd}^{\mathbf{x}^*}) \neq 0, \quad (3.6)$$

where $p_{ij}^{\mathbf{x}^*}$ is the element at the i^{th} row and j^{th} column of $P_{\mathbf{x}^*}$.

Proof. First, we show that the covariance between arbitrary entries of each realization of the product matrix of linearized functions can be non-zero. Afterwards, we will show that this implies that the covariance between arbitrary entries of the product matrix random variate, $P_{\mathbf{x}^*}$ can be non-zero.

Consider a local product matrix constructed as above. Then for each realization of the weight matrices, the product matrix realization $M_i^{(L)}$, defined in the region around \mathbf{x}^* following lemma 1. We can derive the covariance between elements of this product matrix within that region in the same way as in proposition 2, finding similarly that:

$$\begin{aligned}
\hat{\Sigma}_{abcd}^{(3)} = & \alpha_{abcd} \sum_{ij} \left(\delta_{ac} \delta_{ij} \sigma_{ai}^{(3)} \right) \cdot \sum_k \left(\delta_{ij} \delta_{bd} \sigma_{ik}^{(2)} \sigma_{kb}^{(1)} \right. \\
& + \delta_{bd} \mu_{ik}^{(2)} \mu_{jk}^{(2)} \sigma_{kb}^{(1)} \\
& \left. + \delta_{ij} \mu_{kb}^{(1)} \mu_{kd}^{(1)} \sigma_{ik}^{(2)} \right) \\
& + \mu_{ai}^{(3)} \mu_{cj}^{(3)} \cdot \sum_k \left(\delta_{ij} \delta_{bd} \sigma_{ik}^{(2)} \sigma_{kb}^{(1)} \right. \\
& + \delta_{bd} \mu_{ik}^{(2)} \mu_{jk}^{(2)} \sigma_{kb}^{(1)} \\
& \left. + \delta_{ij} \mu_{kb}^{(1)} \mu_{kd}^{(1)} \sigma_{ik}^{(2)} \right) \\
& + \mu_{ib}^{(2)} \mu_{jd}^{(2)} \left(\delta_{ac} \delta_{ij} \sigma_{ai}^{(3)} \right), \tag{A.26}
\end{aligned}$$

where α is a constant determined by the piecewise-linearity in the linear region we are considering. Note that we must assume here that $\alpha_{ij} \neq 0$ except in a region of zero measure, for example a LeakyReLU, otherwise it is possible that the constant introduced by the activation could eliminate non-zero covariances. We discuss this point further below.

Now note that the covariance of the sum of independent random variables is the sum of their covariances. Therefore the covariance of I realizations of P (suppressing the notation (L)) is:

$$\text{Cov}(P_{ab}, P_{cd}) = \frac{1}{I} \sum_{i=1}^I \hat{\Sigma}_{abcd}^i \tag{A.27}$$

As before, consider the case of positive means and standard deviations. Just like before, this results in a positive entry in the covariance between any two elements for each realization of the product matrix, and by eq. (A.27) the entry for any element of the local product matrix remains positive as well. This suffices to prove the theorem for the case of $L = 3$. Just as proposition 2 extends to all larger L , this result does also.

Remark 3. *The above proof assumes that $\alpha_{ij} \neq 0$. In fact, for common piecewise non-linearities like ReLU, α may indeed be zero. This means that the non-linearity can, in principle, ‘disconnect’ regions of the network such*

that the ‘effective depth’ falls below 3 and there is not a covariance between every element.

We cannot rule this out theoretically, as it depends on the data and learned function. In practice, we find it very unlikely that a trained neural network will turn off all its activations for any typical input, nor that enough activations will be zero that the product matrix does not have shared elements after some depth.

However, we do observe that for at least some network structures and datasets it is uncommon in practice that all the activations in several layers are ‘switched off’. We show in fig. 3.5 an example of a local product matrix covariance which does not suffer from this problem. We find that for a model trained with mean-field VI on the FashionMNIST test dataset the number of activations switched on is on average 48.5% with standard deviation 4.7%. There were only four sampled models out of 100 samples on each of 10,000 test points where an entire row of activations was ‘switched off’, reducing the effective depth by one, and this never occurred in more than one row. Indeed, [Goldblum et al., 2019] describe settings with all activations switched off as a pathological case where SGD fails.

□

Existence Proof of a Two-Hidden-Layer Mean-field Approximate Posterior Inducing the True Posterior Predictive

In this section we prove that:

Proposition 1. *Let $p(\mathbf{y} = \mathbf{Y}|\mathbf{x}, \mathcal{D})$ be the probability density function for the posterior predictive distribution of any given multivariate regression function, with $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{y} \in \mathbb{R}^K$, and \mathbf{Y} the posterior predictive random variable. Let $f(\cdot)$ be a Bayesian neural network with two hidden layers. Let $\hat{\mathbf{Y}}$ be the random vector defined by $f(\mathbf{x})$. Then, for any $\epsilon, \delta > 0$, there exists a set of parameters defining the neural network f such that the absolute value of the difference in probability densities for any point is bounded:*

$$\forall \mathbf{y}, \mathbf{x}, i : \Pr \left(|p(y_i = \hat{Y}_i) - p(y_i = Y_i|\mathbf{x}, \mathcal{D})| > \epsilon \right) < \delta, \quad (3.4)$$

so long as: the activations of f are non-polynomial, non-periodic, and have only zero-measure non-monotonic regions, the first hidden layer has at least $D + 1$ units, the second hidden layer has an arbitrarily large number of units, the cumulative density function of the posterior predictive is continuous in output-space, and the probability density function is continuous and finite non-zero everywhere. Here, the probability bound is with respect to the distribution over a subset of the weights described in the proof, θ_{Pr} , while one weight distribution θ_Z remains to induce the random variable $\hat{\mathbf{Y}}$.

Proof. We extend an informal construction by Gal [2016] which aimed to show that a sufficiently deep network with a unimodal approximate posterior could induce a multi-modal posterior predictive by learning the inverse cumulative distribution function (c.d.f.) of the multi-modal distribution. In our case, we are not chiefly interested in the number of modes, but more generally the expressive power of the mean-field distribution in a BNN of sufficient width and depth. First, we outline a simplified version of the proof that highlights the main mechanisms involved but is not constructed with a Bayesian neural network. Later, we prove the full result for Bayesian neural networks.

Simplified Construction

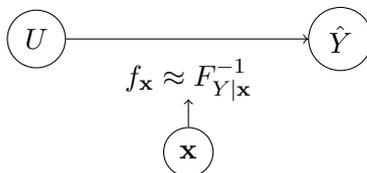


Figure A.2: The random variable whose probability density function is the true predictive posterior $p(y|\mathbf{x}, \mathcal{D})$ can be written $Y|\mathbf{x}$. If it has an inverse cumulative density function (c.d.f.), $F_{Y|\mathbf{x}}^{-1}$, we can transform a uniform random variable U onto it. We can approximate this inverse c.d.f. with $f_{\mathbf{x}}$ indexed by \mathbf{x} . The random variable given by $\hat{Y} := f_{\mathbf{x}}(U)$ can be constructed to be ‘similar’ to $Y|\mathbf{x}$ as we show.

Lemma 2. *Let $p(y = Y|\mathbf{x}, \mathcal{D})$ be the probability density function for the posterior predictive distribution of any given univariate regression function. Let U be a uniformly distributed random variable. Let $f(\cdot)$ be a deterministic neural network with a single hidden layer of arbitrary width and invertible non-polynomial activations. Let \hat{Y} be the random variable defined by $f(U, \mathbf{x})$. Then, for any $\epsilon > 0$, there exists a set of parameters defining a sufficiently wide f such that the absolute value of the difference in probability densities for any point is bounded:*

$$\forall y, \mathbf{x} : |p(y = \hat{Y}) - p(y = Y|\mathbf{x}, \mathcal{D})| < \epsilon, \quad (\text{A.28})$$

so long as the cumulative distribution function of the posterior predictive is continuous in output-space and the probability density function is non-zero everywhere.

Proof. An outline of the proof for the simplified case is shown in Figure A.2.

Suppose there is a true posterior distribution over function outputs whose probability density function (p.d.f) is given by $p(y = Y|\mathbf{x}, \mathcal{D})$. These define a random variable that we will denote $Y|\mathbf{x}$.

We also have some approximation that takes \mathbf{x} as an input and returns some y in the output space. Later, this will be our Bayesian neural network,

but for now we simplify. Instead, our procedure is to have some deterministic neural network which accepts as an input a realization of a uniformly distributed random variable, U , and the input point \mathbf{x} . We define a random variable $\hat{Y} := f(U, \mathbf{x})$.

We would like to show that it is possible to construct a neural network f such that the result in equation A.28 holds—that \hat{Y} is suitably similar to the true predictive posterior random variable $Y|\mathbf{x}$.

First, note that if $Y|\mathbf{x}$ has an inverse cumulative density function (c.d.f.) then, by the universality of the uniform, transforming a uniform random variable by this function creates a random variable distributed as $Y|\mathbf{x}$. As a result, if there is such an invertible cumulative density function, there is also a function mapping U and \mathbf{x} onto $Y|\mathbf{x}$.

Second, consider the conditions under which $Y|\mathbf{x}$ has an invertible c.d.f. We must assume that the c.d.f. is continuous in output space and that the probability density function is non-zero everywhere. The first is reasonable for most normal problems, we often make a stronger assumption of Lipschitz continuity. The second is also relatively mild, corresponding to non-dogmatic certainty (a posterior distribution that puts zero probability density on some output given some input can never update away from that in light of new information). Given these mild assumptions, therefore, we know that there exists a continuous function $F_{Y|\mathbf{x}}^{-1}$ which is the inverse of the c.d.f. of $Y|\mathbf{x}$.

Third, we consider how we might approximate this function. Here, we invoke the universal approximation theorem (UAT) [Leshno et al., 1993]. This states that for any continuous function g , arbitrary fixed error, ϵ , and compact subset \mathcal{A} of \mathbb{R}^D , there exists a deterministic neural network with an arbitrarily wide single layer of hidden units and a non-polynomial activation, f such that:

$$\forall \mathbf{a} \in \mathcal{A} : |f(\mathbf{a}) - g(\mathbf{a})| < \epsilon. \quad (\text{A.29})$$

By setting the arbitrary continuous function as the inverse c.d.f. of $Y|\mathbf{x}$, that is, $g(U, \mathbf{x}) = F_{Y|\mathbf{x}}^{-1}(U)$ (which we have already assumed is continuous) it follows that:

$$\forall u, \mathbf{x} \in \mathcal{A} : |f(u, \mathbf{x}) - F_{Y|\mathbf{x}}^{-1}(u)| < \epsilon, \quad u \sim U. \quad (\text{A.30})$$

Fourth, we convert this bound on the inverse c.d.f. into a bound on the c.d.f. and then a bound on the p.d.f. We rewrite the function represented by the neural network to make explicit that we are using \mathbf{x} to index a function from u to y : $y = f(u, \mathbf{x}) = f_{\mathbf{x}}(u)$.

For this step, we will need to be able to invert the approximation to the inverse CDF such that $u = f_{\mathbf{x}}^{-1}(y)$ (for fixed \mathbf{x}). In general for neural network functions this is not true. As a result, we employ a construction which breaks apart the line over which y runs into subsegments within which the network is invertible. For non-periodic activation functions which have only zero-measure non-monotonic regions (e.g., ReLU) there will be finitely many of these segments given a finite number of hidden units. Let us index over

these subregions with i , noting that we can think of the distribution over y as a weighted mixture distribution whose members have zero-density outside of the subregions. The approximate inverse CDF of each of these sub-region mixture members can be written as $f_{\mathbf{x}}^i(u)$ such that $f_{\mathbf{x}}(u) = \sum_i f_{\mathbf{x}}^i(u)$. Each of these $f_{\mathbf{x}}^i(u)$ is invertible. We can therefore rewrite equation A.30 as:

$$\forall y, \mathbf{x} \in \mathcal{A} : \quad \left| \sum_i f_{\mathbf{x}}(f_{\mathbf{x}}^{i-1}(y)) - F_{Y|\mathbf{x}}^{-1}(f_{\mathbf{x}}^{i-1}(y)) \right| < \epsilon. \quad (\text{A.31})$$

which implies that:

$$\forall y, \mathbf{x} \in \mathcal{A} : \quad \left| \sum_i y - F_{Y|\mathbf{x}}^{-1}(f_{\mathbf{x}}^{i-1}(y)) \right| < \epsilon. \quad (\text{A.32})$$

Remember that we assumed above that the c.d.f. of $Y|\mathbf{x}$ is uniformly continuous, which means that for any y' and y'' , and any $\epsilon > 0$ there exists a δ such that if $|y' - y''| < \delta$ then $|F_{Y|\mathbf{x}}(y') - F_{Y|\mathbf{x}}(y'')| < \epsilon$. Alongside equation A.32, and canceling the c.d.f. with the inverse c.d.f. this entails that:

$$\forall y, \mathbf{x} \in \mathcal{A} : \quad \left| \sum_i F_{Y|\mathbf{x}}(y) - f_{\mathbf{x}}^{i-1}(y) \right| < \epsilon. \quad (\text{A.33})$$

But since $f_{\mathbf{x}}^{i-1}(y)$ is zero by construction outside of its subregion, this results in a bound on the overall c.d.f. of the random variable. That is to say, the bound on the inverse c.d.f. implies a bound on the c.d.f.

Finally, we remember that the cumulative density is the integral of the probability density function. Therefore, by Theorem 7.17 of Fedorov [1972] and the uniform convergence in the c.d.f.s, it follows that:

$$\forall y, \mathbf{x} \in \mathcal{A} : \quad |p(y = Y|\mathbf{x}, \mathcal{D}) - p(y = \hat{Y})| < \epsilon \quad (\text{A.34})$$

introducing a bound in the probability density functions of the random variable of true posterior outputs and the outputs of the approximation $\hat{Y} = f(U, \mathbf{x})$. \square

Full Construction

The full construction extends the result above in the following ways:

- Rather than separately introducing U , we show how the first layer of a Bayesian neural network can map $\mathbf{x} \rightarrow \mathbf{x}', Z$, where Z is a unit Gaussian random variable and \mathbf{x}' is a noised version of \mathbf{x} .
- Rather than using a deterministic neural network for the universal approximation theorem, we apply the stochastic adaptation introduced by Foong et al. [2020].

- Rather than a univariate regression, we consider multivariate regression.

Like Leshno et al. [1993] we note that the extension from univariate to multivariate regression follows trivially from the existence of a mapping from $\mathbb{R} \rightarrow \mathbb{R}^K$.⁴

We first give some intuition as to how the proof works. The first weight layer serves to map $\mathbf{x} \rightarrow \mathbf{x}', Z$. This sets us up in a similar situation to the proof in the previous section, where we began with \mathbf{x}, U . This requires two small adjustments to the proof above. The first is that the random variable we introduce is now Gaussian, rather than Uniform. The second is that all our results will be in terms of \mathbf{x}' , rather than \mathbf{x} , and an additional step will be required to convert a probabilistic bound in one to the other (noting that we can freely set the weights in the first layer to have arbitrarily small variance).

The second weight layer will play the role of the neural network in the simplified proof. This also will require a small modification, because earlier we assumed that the neural network was deterministic, but it is now stochastic. This means that the final result becomes a probabilistic bound.

As a result of all of these changes, the proof becomes considerably more complicated, though nothing important changes in the intuition behind the construction.

Step 1: Mapping $\mathbf{x} \rightarrow \mathbf{x}', Z$

Consider inputs $\mathbf{x} \in \mathbb{R}^D$. We define a two-hidden-layer neural network with an invertible non-polynomial activation function $\phi : \mathbb{R} \mapsto \mathbb{R}$. The first component of the network is a single weight matrix mapping onto a vector of hidden units: $\mathbf{h}_1 = \phi(\boldsymbol{\theta}_0 \mathbf{x} + \mathbf{b}_0)$. The second component is a neural network with a layer of hidden units defined relative to the first layer of units $\mathbf{h}_2 = \phi(\boldsymbol{\theta}_1 \mathbf{h}_1 + \mathbf{b}_1)$, and outputs $\mathbf{y} = \boldsymbol{\theta}_2 \mathbf{h}_2 + \mathbf{b}_2$. The distribution over the outputs \mathbf{y} defines the random variable $\hat{\mathbf{Y}}$. Here, $\boldsymbol{\theta}_0$, $\boldsymbol{\theta}_1$, and $\boldsymbol{\theta}_2$ are matrices of independent Gaussian random variables and \mathbf{b}_0 , \mathbf{b}_1 , and \mathbf{b}_2 are vectors of independent Gaussian random variables. Given some dataset \mathcal{D} the predictive posterior distribution over outputs is $p(\mathbf{y}|\mathbf{x}, \mathcal{D})$ which we associate with the random variable \mathbf{Y} . This is our (intractable) target.

Consider only the first component of the neural network, which maps \mathbf{x} onto \mathbf{h}_1 . We can construct simple constraints on $\boldsymbol{\theta}_0$ and \mathbf{b}_0 such that:

$$\mathbf{h}_1 = \begin{pmatrix} \phi(z) \\ \phi(\mathbf{x}') \end{pmatrix}, \quad (\text{A.35})$$

where $z \sim Z$, a unit Gaussian random variable, and $\mathbf{x}' \in \mathbb{R}^D$, such that $\Pr(\|\mathbf{x}' - \mathbf{x}\| > \epsilon_1) < \delta_1$. In particular, suppose that for $\boldsymbol{\theta}_0 \in \mathbb{R}^{D \times D+1}$ and

⁴A slightly complication is added by the continuity requirements. However, we note that the assumption that the p.d.f. is finite everywhere guarantees that there is a continuous function over y which contains continuous segments for each of K dimensions, even if those individual segments are not continuous with each other.

$\mathbf{b}_0 \in \mathbb{R}^{D+1}$.

$$\boldsymbol{\theta}_0 = \mathcal{N}(M_{\boldsymbol{\theta}_0}, \Sigma_{\boldsymbol{\theta}_0}) \text{ where } M_{\boldsymbol{\theta}_0} = (\mathbf{0}_D, \mathbb{I}_{D \times D}); \Sigma_{\boldsymbol{\theta}_0} = \sigma^2 \mathbb{I}_{D \times D+1}; \quad (\text{A.36})$$

$$\mathbf{b}_0 = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{b}_0}, \boldsymbol{\sigma}_{\mathbf{b}_0}) \text{ where } \boldsymbol{\mu}_{\mathbf{b}_0} = \mathbf{0}_{D+1}, \boldsymbol{\sigma}_{\mathbf{b}_0} = (1 \ \sigma \ \dots \ \sigma). \quad (\text{A.37})$$

By multiplication, straightforwardly $\mathbf{x}' = \mathcal{N}(\mathbf{x}, 2\sigma^2 \mathbf{1})$. It follows trivially that for any ϵ_1 and δ_1 there exists some σ such that the bound holds. We will apply this bound at the end of the proof to convert a bound in \mathbf{x}' to one in \mathbf{x} .

Here, we introduce a distinction between the weights which determine \mathbf{x}' and those that create Z . The only weights which determine Z are the first element of $\boldsymbol{\theta}_0$ and the first element of \mathbf{b}_0 . Call these $\boldsymbol{\theta}_Z$. We then define the remainder of the weight distributions as $\boldsymbol{\theta}_{\text{Pr}} := \{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2\} \setminus \boldsymbol{\theta}_Z$. This distinction is important, because the probabilistic bound in the proof will be over $\boldsymbol{\theta}_{\text{Pr}}$ while the distribution over $\boldsymbol{\theta}_Z$ will induce the random variable $\hat{\mathbf{Y}}$.

Step 2: Invoking a Result Similar to the Simplified Construction We show that there is a function which maps Z and \mathbf{x}' onto \mathbf{Y} , under reasonable assumptions similar to those of the simplified construction. For brevity, we denote the probability density function (p.d.f.) of the true posterior predictive distribution of the random variable \mathbf{Y} conditioned on \mathbf{x}' and \mathcal{D} as $f_{Y|\mathbf{x}'}$ $\equiv p(Y = y|\mathbf{x}', \mathcal{D})$ and the cumulative density function (c.d.f.) as $F_{Y|\mathbf{x}'}$. We similarly write the inverse c.d.f. as $F_{Y|\mathbf{x}'}^{-1}$.

We must adapt the simplified construction to account for the fact that rather than simply approximating the inverse of the c.d.f. we now need to also transform the Gaussian random variable onto a Uniform one and invert the activation. We show below:

Lemma 3. *There exists continuous function $G_{\mathbf{x}'}^{-1} = F_{Y|\mathbf{x}'}^{-1} \cdot F_Z \cdot \phi^{-1}$, where F_Z is the c.d.f. of the unit Gaussian and ϕ^{-1} is the inverse of the activation function, such that the random variable $G_{\mathbf{x}'}^{-1}(\phi(Z))$ is equal in distribution to $\mathbf{Y}|\mathbf{x}'$, if the p.d.f. of the posterior predictive is non-zero everywhere and the c.d.f. is continuous.*

The limitation to p.d.f.s is modest as before. The naming of the function $G_{\mathbf{x}'}^{-1}$ is suggestive, and indeed its inverse exists if $F_{Y|\mathbf{x}'}$ is invertible, since the c.d.f. of a unit Gaussian is invertible (though this function cannot be easily expressed).

Whereas in the simplified construction we showed that the neural network could approximate the inverse c.d.f., here we show that the second hidden layer of our larger Bayesian neural network can approximate the more complicated function required by lemma 3. This allows the second hidden layer of the neural network to transform \mathbf{x}', Z onto $\hat{\mathbf{Y}}$ such that $\hat{\mathbf{Y}}$ is appropriately similar to \mathbf{Y} . First we show that we can approximate the function $G_{\mathbf{x}'}^{-1}$, generating a probabilistic bound because the weights of the neural network are now Gaussian random variables:

Lemma 4. For a uniformly continuous function $G_{\mathbf{x}'}^{-1}(z) : z, \mathbf{x}' \mapsto \mathbf{y}$, for any $\epsilon, \delta > 0$ and compact subset \mathcal{A} of \mathbb{R}^D , there exist fully-factorized Gaussian approximating distributions $q(\boldsymbol{\theta}_1)$, $q(\boldsymbol{\theta}_2)$, $q(\mathbf{b}_1)$, and $q(\mathbf{b}_2)$, and a function over the outputs of the later part of the neural network: $\hat{G}^{-1}(\mathbf{x}', z) \equiv \boldsymbol{\theta}_2(\sigma(\boldsymbol{\theta}_1 \mathbf{h}_1) + \mathbf{b}_1) + \mathbf{b}_2$ (remembering that $\mathbf{h}_1 \equiv \phi(z, \mathbf{x}')$), such that:

$$\Pr\left(|\hat{G}^{-1}(\mathbf{x}', z) - G_{\mathbf{x}'}^{-1}(z)| > \epsilon\right) < \delta, \quad \forall \mathbf{x}', z, \mathcal{A}. \quad (\text{A.38})$$

The probability measure is over the weight distributions of $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathbf{b}_1, \mathbf{b}_2$.

Having shown that the second component can approximate the inverse c.d.f. to within a bound, we as before we further show that the random variable created by this transformation has a p.d.f. within a bound of the p.d.f. of \mathbf{Y} , which suffices to prove the desired result.

For this, we show this below for the transformed variable \mathbf{x}' :

Lemma 5. For any $\epsilon > 0$ and $\delta > 0$ there exists a mean-field weight distribution $q(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathbf{b}_1, \mathbf{b}_2)$ such that the probability density functions are bounded:

$$\Pr\left(|p(y_i = \hat{\mathbf{Y}}_i) - p(y_i = \mathbf{Y}_i | \mathbf{x}', \mathcal{D})| > \epsilon\right) < \delta, \quad \forall \mathbf{x}', y_i, \mathcal{A}. \quad (\text{A.39})$$

We then move the bounds onto an expression in the original features, \mathbf{x} . Recall from before that because the variance of the weights in the first layer can be arbitrarily small, that for any ϵ there is a δ :

$$\Pr\left(\|\mathbf{x}' - \mathbf{x}\| > \epsilon\right) < \delta \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{A}, \quad (\text{A.40})$$

where the probability measure is over $\boldsymbol{\theta}_{\text{Pr}}$. Moreover, since we have assumed that the probability density function is continuous, this bound alongside the previous bound on the probability density functions jointly entail that:

$$\Pr\left(|p(y_i = \hat{\mathbf{Y}}_i) - p(y_i = \mathbf{Y}_i | \mathbf{x}, \mathcal{D})| > \epsilon\right) < \delta, \quad \forall \mathbf{x} \in \mathcal{A}, y_i. \quad (\text{A.41})$$

where the probability measure is over $\boldsymbol{\theta}_{\text{Pr}}$. □

Below, we prove the lemmas required in the proposition above.

Lemma 3. There exists continuous function $G_{\mathbf{x}'}^{-1} = F_{\mathbf{Y}|\mathbf{x}'}^{-1} \cdot F_Z \cdot \phi^{-1}$, where F_Z is the c.d.f. of the unit Gaussian and ϕ^{-1} is the inverse of the activation function, such that the random variable $G_{\mathbf{x}'}^{-1}(\phi(Z))$ is equal in distribution to $\mathbf{Y}|\mathbf{x}$, if the p.d.f. of the posterior predictive is non-zero everywhere and the c.d.f. is continuous.

Proof. Trivially $\phi^{-1}(\phi(Z)) = Z$.

Let F_Z be the cumulative distribution function (c.d.f.) of the unit Gaussian random variable Z . By the Universality of the Uniform, $U = F_Z(Z)$ has a standard uniform distribution.

Let $F_{Y|\mathbf{x}}$ be the c.d.f. of \mathbf{Y} conditioned on \mathbf{x} , and \mathcal{D} . Suppose that the posterior predictive is non-zero everywhere (that is, you cannot rule out that there's even the remotest chance of any y_i given some input \mathbf{x} , however small). Then, since the c.d.f. is continuous by assumption, $F_{Y|\mathbf{x}}$ is invertible.

Again, by the Universality of the Uniform $U' = F_{Y|\mathbf{x}}(y)$ has a standard uniform distribution. So $\forall u : p(U = u) = p(U' = u)$. Moreover $F_{Y|\mathbf{x}}$ is invertible. So $p(Y = y) = F_{Y|\mathbf{x}}^{-1}(F_Z(Z))$.

It follows that there exists a continuous function as required. \square

Lemma 4. *For a uniformly continuous function $G_{\mathbf{x}'}^{-1}(z) : z, \mathbf{x}' \mapsto \mathbf{y}$, for any $\epsilon, \delta > 0$ and compact subset \mathcal{A} of \mathbb{R}^D , there exist fully-factorized Gaussian approximating distributions $q(\boldsymbol{\theta}_1)$, $q(\boldsymbol{\theta}_2)$, $q(\mathbf{b}_1)$, and $q(\mathbf{b}_2)$, and a function over the outputs of the later part of the neural network: $\hat{G}^{-1}(\mathbf{x}', z) \equiv \boldsymbol{\theta}_2(\sigma(\boldsymbol{\theta}_1 \mathbf{h}_1) + \mathbf{b}_1) + \mathbf{b}_2$ (remembering that $\mathbf{h}_1 \equiv \phi(z, \mathbf{x}')$), such that:*

$$\Pr\left(|\hat{G}^{-1}(\mathbf{x}', z) - G_{\mathbf{x}'}^{-1}(z)| > \epsilon\right) < \delta, \quad \forall \mathbf{x}', z, \mathcal{A}. \quad (\text{A.38})$$

The probability measure is over the weight distributions of $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathbf{b}_1, \mathbf{b}_2$.

Proof. The Universal Approximation Theorem (UAT) states that for any continuous function f , and an arbitrary fixed error, e , and compact subset \mathcal{A} of \mathbb{R}^D , there exists a deterministic neural network with an arbitrarily wide single layer of hidden units and a non-polynomial activation, σ :

$$\forall \mathbf{x} \in \mathcal{A} : |\sigma(\mathbf{w}_2(\sigma(\mathbf{w}_1 \mathbf{x}) + \mathbf{b}_1) + \mathbf{b}_2) - f(\mathbf{x})| < e. \quad (\text{A.42})$$

In addition, we make use of Lemma 7 of [Foong et al., 2020]. This states that for any $e', \delta_2 > 0$, for some fixed means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_{b_1}, \boldsymbol{\mu}_{b_2}$ of $q(\boldsymbol{\theta}_1)$, $q(\boldsymbol{\theta}_2)$, $q(\mathbf{b}_1)$, and $q(\mathbf{b}_2)$ respectively, there exists some standard deviation $s' > 0$ for all those approximate posteriors such that for all $s < s'$, for any $\mathbf{h}_1 \equiv (z, \mathbf{x}') \in \mathbb{R}^{N+1}$

$$\Pr\left(|\sigma(\boldsymbol{\theta}_2(\sigma(\boldsymbol{\theta}_1 \mathbf{h}_1) + \mathbf{b}_1) + \mathbf{b}_2) - \sigma(\boldsymbol{\mu}_2(\sigma(\boldsymbol{\mu}_1 \mathbf{h}_1) + \boldsymbol{\mu}_{b_1}) + \boldsymbol{\mu}_{b_2})| > e'\right) < d. \quad (\text{A.43})$$

Note that the deterministic weights of eq. (A.42) can just be these means. As a result:

$$\Pr\left(|\sigma(\boldsymbol{\theta}_2(\sigma(\boldsymbol{\theta}_1 \mathbf{h}_1) + \mathbf{b}_1) + \mathbf{b}_2) - f(\mathbf{h}_1)| > e + e'\right) < \delta. \quad (\text{A.44})$$

We note that we define $\hat{G}^{-1}(\mathbf{x}', z) \equiv \sigma(\boldsymbol{\theta}_2(\sigma(\boldsymbol{\theta}_1 \mathbf{h}_1) + \mathbf{b}_1) + \mathbf{b}_2)$ as above, and that $f(\mathbf{h}_1)$ may be $G_{\mathbf{x}'}^{-1}(z)$, which is assumed to be uniformly continuous. It follows, allowing $\epsilon = e + e'$:

$$\Pr\left(|\hat{G}^{-1}(\mathbf{x}', z) - G_{\mathbf{x}'}^{-1}(z)| > \epsilon\right) < \delta. \quad (\text{A.45})$$

as required.

Lemma 5. For any $\epsilon > 0$ and $\delta > 0$ there exists a mean-field weight distribution $q(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathbf{b}_1, \mathbf{b}_2)$ such that the probability density functions are bounded:

$$\Pr\left(\left|p(y_i = \hat{\mathbf{Y}}_i) - p(y_i = \mathbf{Y}_i | \mathbf{x}', \mathcal{D})\right| > \epsilon\right) < \delta, \quad \forall \mathbf{x}', y_i, \mathcal{A}. \quad (\text{A.39})$$

In this lemma, we show that a bound on the inverse c.d.f. used to map Z onto our target implies a bound in the p.d.f. of that constructed random variable to the p.d.f. of our target. This lemma follows the argument of the simplified construction, with some additional complexity of notation introduced by the requirement that the input random variable was Gaussian rather than Uniform. Here, we complete the proof steps with a univariate y to simplify notation, noting that because we can map $\mathbb{R} \rightarrow \mathbb{R}^K$ the multivariate regression follows trivially from the univariate result.

We first note that the result of lemma 4 can be applied to $z' = \hat{G}(\mathbf{x}', y)$ using an inverted version of our network function such that:

$$\Pr\left(\left|\hat{G}^{-1}(\mathbf{x}', \hat{G}(\mathbf{x}', y)) - G_{\mathbf{x}'}^{-1}(\hat{G}(\mathbf{x}', y))\right| > \epsilon_2\right) < \delta_2, \quad \forall \mathbf{x}' \in \mathcal{A}, y. \quad (\text{A.46})$$

We further note that by the triangle inequality:

$$\left|G_{\mathbf{x}'}^{-1}(G_{\mathbf{x}'}(y)) - G_{\mathbf{x}'}^{-1}(\hat{G}(\mathbf{x}', y))\right| \leq \left|G_{\mathbf{x}'}^{-1}(G_{\mathbf{x}'}(y)) - \hat{G}^{-1}(\mathbf{x}', \hat{G}(\mathbf{x}', y))\right| \quad (\text{A.47})$$

$$+ \left|\hat{G}^{-1}(\mathbf{x}', \hat{G}(\mathbf{x}', y)) - G_{\mathbf{x}'}^{-1}(\hat{G}(\mathbf{x}', y))\right|, \quad (\text{A.48})$$

and since $G_{\mathbf{x}'}^{-1}(G_{\mathbf{x}'}(y)) = \hat{G}^{-1}(\mathbf{x}', \hat{G}(\mathbf{x}', y)) = \mathbf{x}'$:

$$\leq \left|\hat{G}^{-1}(\mathbf{x}', \hat{G}(\mathbf{x}', y)) - G_{\mathbf{x}'}^{-1}(\hat{G}(\mathbf{x}', y))\right|. \quad (\text{A.49})$$

Inserting this inequality into the result of lemma 4, we have that:

$$\Pr\left(\left|G_{\mathbf{x}'}^{-1}(G_{\mathbf{x}'}(y)) - G_{\mathbf{x}'}^{-1}(\hat{G}(\mathbf{x}', y))\right| > \epsilon_2\right) < \delta_2, \quad \forall \mathbf{x}' \in \mathcal{A}, y. \quad (\text{A.50})$$

But we further note that we have assumed that the c.d.f. $G_{\mathbf{x}'}$ is uniformly continuous so for any y' and y'' , and for any $\epsilon' > 0$ there is an $\epsilon'' > 0$ and vice versa, such that if:

$$|y' - y''| < \epsilon', \quad (\text{A.51})$$

then:

$$|G_{\mathbf{x}'}(y') - G_{\mathbf{x}'}(y'')| < \epsilon''. \quad (\text{A.52})$$

It follows that for any ϵ and δ there is a $q(\boldsymbol{\theta})$ such that:

$$\Pr\left(\left|G_{\mathbf{x}'}(G_{\mathbf{x}'}^{-1}(G_{\mathbf{x}'}(y))) - G_{\mathbf{x}'}(G_{\mathbf{x}'}^{-1}(\hat{G}(\mathbf{x}', y)))\right| > \epsilon\right) < \delta, \quad \forall \mathbf{x}' \in \mathcal{A}, y, \quad (\text{A.53})$$

and therefore:

$$\Pr\left(\left|G_{\mathbf{x}'}(y) - \hat{G}(\mathbf{x}', y)\right| > \epsilon\right) < \delta, \quad \forall \mathbf{x}' \in \mathcal{A}, y. \quad (\text{A.54})$$

Next, we remember that $G_{\mathbf{x}'} = \phi \cdot F_Z^{-1} \cdot F_{Y|\mathbf{x}'}$, and that ϕ and F_Z^{-1} are continuous, and therefore:

$$\Pr\left(\left|F_{Y|\mathbf{x}'}(y) - F_{\hat{Y}|\mathbf{x}'}(y)\right| > \epsilon\right) < \delta, \quad \forall \mathbf{x}' \in \mathcal{A}, y, \quad (\text{A.55})$$

where $F_{\hat{Y}|\mathbf{x}'}(y) = F_Z(\phi^{-1}(\hat{G}(\mathbf{x}', y)))$.

As a final step, we remember that the cumulative density is the integral of the probability density function. Therefore, by Theorem 7.17 of Fedorov [1972] and the uniform convergence in the c.d.f.s, it follows that there for any bounds there exists $q(\boldsymbol{\theta})$ such that:

$$\Pr\left(\left|f_{\hat{Y}|\mathbf{x}'} - f_{Y|\mathbf{x}'}\right| > \epsilon\right) < \delta, \quad \forall \mathbf{x}' \in \mathcal{A}, \quad (\text{A.56})$$

Writing out the probability density functions fully and mapping the univariate function the multivariate we have:

$$\Pr\left(\left|p(y_i = \hat{\mathbf{Y}}_i) - p(y_i = \mathbf{Y}_i|\mathbf{x}', \mathcal{D})\right| > \epsilon\right) < \delta, \quad \forall \mathbf{x}' \in \mathcal{A}, y_i. \quad (\text{A.57})$$

□

Appendix B

Appendix to Approximation Assumptions Affect Optimization

B.1 Derivation of the Entropy Term of the KL-divergence

In this section, we show that the component of KL-divergence term of the loss which is the entropy of the posterior distribution over the weights $q(\mathbf{w}^{(x)})$ can be estimated as:

$$\mathcal{L}_{\text{entropy}} := \int q(\mathbf{w}^{(x)}) \log[q(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)} \quad (\text{B.1})$$

$$= - \sum_i \log[\sigma_i^{(x)}] + \text{const} \quad (\text{B.2})$$

where i is an index over the weights of the model.

Throughout this section we use a superscript indicates the basis—an (x) means we are in the Cartesian coordinate system tied to the weight-space while (r) is the hyperspherical coordinate system (the letter is the canonical ‘first’ coordinate of that coordinate system).

We begin by applying the reparameterization trick [Kingma et al., 2014, Rezende et al., 2014]. Following the auxiliary variable formulation of Gal [2016], we express the probability density function of $q(\mathbf{w}^{(x)})$ with an auxiliary

variable.

$$q(\mathbf{w}^{(x)}) = \int q(\mathbf{w}^{(x)}, \boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)} \quad (\text{B.3})$$

$$= \int q(\mathbf{w}^{(x)} | \boldsymbol{\epsilon}^{(r)}) q(\boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)} \quad (\text{B.4})$$

$$= \int \delta(\mathbf{w}^{(x)} - g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})) q(\boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)}. \quad (\text{B.5})$$

In equation (B.5), we have used a reparameterization trick transformation:

$$g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \mathbf{T}_{rx}(\boldsymbol{\epsilon}^{(r)}) \quad (\text{B.6})$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are parameters of the model and where \mathbf{T}_{rx} is the standard transformation from hyperspherical into Cartesian coordinates.

Substituting equation (B.5) into the definition of the entropy loss term in equation (B.1), and applying the definition of the Kronecker delta we can eliminate dependence on $\mathbf{w}^{(x)}$:

$$\mathcal{L}_{\text{entropy}} = \int q(\mathbf{w}^{(x)}) \log[q(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)} \quad (\text{B.7})$$

$$= \int \left(\int \delta(\mathbf{w}^{(x)} - g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})) q(\boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)} \right) \log[q(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)} \quad (\text{B.8})$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log[q(g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)}))] d\boldsymbol{\epsilon}^{(r)}. \quad (\text{B.9})$$

Then, we perform a coordinate transformation from $g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})$ to $\boldsymbol{\epsilon}^{(r)}$ using the Jacobian of the transformation and simplify.

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[q(\boldsymbol{\epsilon}^{(r)}) \left| \frac{\partial g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})}{\partial \boldsymbol{\epsilon}^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \quad (\text{B.10})$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[q(\boldsymbol{\epsilon}^{(r)}) \left| \prod_i \sigma_i^{(x)} \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \quad (\text{B.11})$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[q(\boldsymbol{\epsilon}^{(r)}) \left| \text{diag}(\boldsymbol{\sigma}) \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \quad (\text{B.12})$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[\frac{q(\boldsymbol{\epsilon}^{(r)})}{\prod_i \sigma_i^{(x)}} \left| \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \quad (\text{B.13})$$

In the last line we have used the fact that $\forall i : \sigma_i^{(x)} \geq 0$ allowing us to pull the determinant of this diagonal matrix out. $\left| \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|$ is the determinant of the

Jacobian for the transformation from Cartesian to hyperspherical coordinates for which we use the result by Muleshkov and Nguyen [2016]:

$$\left| \frac{\partial \boldsymbol{\epsilon}_i^{(x)}}{\partial \boldsymbol{\epsilon}_j^{(r)}} \right| = \text{abs} \left((-1)^{D-1} (\epsilon_0^{(r)})^{D-1} \prod_{i=2}^{D-1} (\sin(\epsilon_i^{(r)}))^{i-1} \right). \quad (\text{B.14})$$

We know that $\epsilon_0^{(r)} \geq 0$ because the radial dimension in hyperspherical coordinates can be assumed positive without loss of generality. We also know $0 \leq \epsilon_i^{(r)} \leq \pi$ for $2 \leq i \leq D-1$ for the hyperspherical coordinate system. So we can simplify the signs:

$$= (\epsilon_0^{(r)})^{D-1} \prod_{i=2}^{D-1} (\sin(\epsilon_i^{(r)}))^{i-1}. \quad (\text{B.15})$$

Therefore, plugging equation (B.15) into (B.13):

$$\mathcal{L}_{\text{entropy}} = \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[\frac{q(\boldsymbol{\epsilon}^{(r)})}{\text{abs}(\prod_i \sigma_i^{(x)})} \left| \frac{\partial \boldsymbol{\epsilon}_i^{(x)}}{\partial \boldsymbol{\epsilon}_j^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \quad (\text{B.16})$$

$$\begin{aligned} &= \int q(\boldsymbol{\epsilon}^{(r)}) \log[q(\boldsymbol{\epsilon}^{(r)})] \\ &\quad - \log[\text{abs}(\prod_i \sigma_i^{(x)})] \\ &\quad - \log \left[(\epsilon_0^{(r)})^{D-1} \prod_{i=2}^{D-1} (\sin(\epsilon_i^{(r)}))^{i-1} \right] d\boldsymbol{\epsilon}^{(r)}. \end{aligned} \quad (\text{B.17})$$

Only the middle term depends on the parameters, and we must therefore only compute this term in order to compute gradients. For sake of completeness, we address the other integrals below, in case one wants to have the full value of the loss (though since it is a lower bound in any case, the full value is not very useful).

The probability density function of the noise variable is separable into independent distributions. The distribution of $\epsilon_0^{(r)}$ is a unit Gaussian. The angular dimensions are distributed so that sampling is uniform over the hypersphere. However, this does **not** mean that the distribution over each angle is uniform, as this would lead to bunching near the n-dimensional generalization of the poles. (Intuitively, there is more surface area per unit of angle near the equator, as is familiar from cartography.) Instead, we use the fact that the area element over the hypersphere is:

$$dA = (\epsilon_0^{(r)})^{D-1} \prod_{i=1}^{D-1} \sin(\epsilon_i^{(r)})^{i-1} d\epsilon_i^{(r)} \quad (\text{B.18})$$

where we remember that $\epsilon_1^{(r)}$ is between $-\pi$ and π , and the rest of the angular elements of $\epsilon^{(r)}$ are between 0 and π . The resulting probability density function is:

$$q(\epsilon^{(r)}) = \prod_{i=0}^D q(\epsilon_i^{(r)}) = \left(\epsilon_0^{(r)}\right)^{D-1} \frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}} \cdot \prod_{i=2}^{D-1} \sin(\epsilon_i^{(r)})^{i-1}. \quad (\text{B.19})$$

As a result, equation (B.17) becomes analytically tractable. Using equation (B.19) we have that

$$\log [q(\epsilon^{(r)})] - \log \left[\left(\epsilon_0^{(r)}\right)^{D-1} \prod_{i=2}^{D-1} \left(\sin(\epsilon_i^{(r)})\right)^{i-1} \right] = -\frac{1}{2} \log 2\pi - \frac{(\epsilon_0^{(r)})^2}{2} \quad (\text{B.20})$$

where the second two terms cancel with the third term in equation (B.17). That is, the first and third terms of equation (B.17) mostly cancel. That is,

$$\mathcal{L}_{\text{entropy}} = - \int q(\epsilon^{(r)}) \left(\log[\prod_i \sigma_i^{(x)}] + \frac{1}{2} \log 2\pi + \frac{(\epsilon_0^{(r)})^2}{2} \right) d\epsilon^{(r)}. \quad (\text{B.21})$$

$$= - \log[\prod_i \sigma_i^{(x)}] - \frac{1}{2} \log 2\pi - \int_0^\infty q(\epsilon_0^{(r)}) \frac{(\epsilon_0^{(r)})^2}{2} d\epsilon_0^{(r)} \quad (\text{B.22})$$

which, remembering that the radial dimension is distributed as a doubled unit normal (adjusting for being strictly positive) leaves us with

$$= - \sum_i \log[\sigma_i^{(x)}] - \frac{1}{2} \log 2\pi - \frac{1}{2}. \quad (\text{B.23})$$

And because we are optimizing the loss and can generally neglect constant terms

$$\mathcal{L}_{\text{entropy}} = - \sum_i \log[\sigma_i^{(x)}] + \text{const.} \quad (\text{B.24})$$

B.2 Setting a Radial Prior

In most of our experiments, we use a typical multivariate Gaussian unit prior in order to ensure comparability with prior work. However, in some settings, such as the Variational Continual Learning setting, it is useful to use the radial posterior as a prior. In these cases, the entropy term is identical to the term for a Gaussian prior. The cross-entropy term can, in principle, be found using Monte Carlo estimation just like the Gaussian case. However, this requires us to compute the expectation of the log of the probability density function. While the probability density function is easy to write down in

hyperspherical coordinates, it is not trivial to state for Cartesian coordinates. This means that we must complete a change of variables in order to compute the probability density function in hyperspherical coordinates

We begin similarly to the previous derivation, with all unchanged expect that we are estimating

$$\mathcal{L}_{\text{cross-entropy}} = \int q(\mathbf{w}^{(x)}) \log[p(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)}. \quad (\text{B.25})$$

In addition to our coordinate transformation $g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})$, we also define the probability density function over the prior with a similar transformation

$$g_{\text{prior}}(\boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\sigma}_{\text{prior}}, \boldsymbol{\epsilon}^{(r)}) = \boldsymbol{\mu}_{\text{prior}} + \boldsymbol{\sigma}_{\text{prior}} \cdot \mathbf{T}_{\text{rx}}(\boldsymbol{\epsilon}^{(r)}) \text{ with } \boldsymbol{\epsilon}^{(r)} \sim p_{\epsilon}(\boldsymbol{\epsilon}^{(r)}) = q_{\epsilon}(\boldsymbol{\epsilon}^{(r)}). \quad (\text{B.26})$$

We begin similarly to the derivation for the entropy loss above

$$\mathcal{L}_{\text{cross-entropy}} = \int q(\mathbf{w}^{(x)}) \log[p(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)} \quad (\text{B.27})$$

$$= \int \left(\int \delta(\mathbf{w}^{(x)} - g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})) q(\boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)} \right) \log[p(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)} \quad (\text{B.28})$$

$$= \int q_{\epsilon}(\boldsymbol{\epsilon}^{(r)}) \log[p(g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)}))] d\boldsymbol{\epsilon}^{(r)}. \quad (\text{B.29})$$

Then, we perform a coordinate transformation which is, unlike the previous case, from $g_{\text{prior}}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})$ to $\boldsymbol{\epsilon}^{(r)}$. By the change of variables formula

$$p(\mathbf{w}^{(x)}) = p_{\epsilon}(g_{\text{prior}}^{-1}(\mathbf{w}^{(x)})) \left| \det \left[\frac{\partial g_{\text{prior}}^{-1}(\mathbf{w}^{(x)})}{\partial \mathbf{w}^{(x)}} \right] \right|, \quad (\text{B.30})$$

and then by the inverse function theorem

$$= p_{\epsilon}(g_{\text{prior}}^{-1}(\mathbf{w}^{(x)})) \left| \det \left[\frac{\partial g_{\text{prior}}(\boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\sigma}_{\text{prior}}, \boldsymbol{\epsilon}^{(r)})}{\partial \boldsymbol{\epsilon}^{(r)}} \right]^{-1} \right|, \quad (\text{B.31})$$

and by the chain rule

$$= p_{\epsilon}(g_{\text{prior}}^{-1}(\mathbf{w}^{(x)})) \left| \det \left[\frac{\partial g_{\text{prior}}(\boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\sigma}_{\text{prior}}, \boldsymbol{\epsilon}^{(r)})}{\partial \boldsymbol{\epsilon}^{(x)}} \frac{\partial \boldsymbol{\epsilon}^{(r)}}{\partial \boldsymbol{\epsilon}^{(x)}} \right]^{-1} \right|. \quad (\text{B.32})$$

Using our earlier results from eq. (B.15) and eq. (B.13) this is equal to

$$= p_{\epsilon}(g_{\text{prior}}^{-1}(\mathbf{w}^{(x)})) \left| (\epsilon_0^{(r)})^{D-1} \prod_{i=2}^D (\sin(\epsilon_i^{(r)}))^{i-1} \prod_j \sigma_{\text{prior},j} \right|^{-1}. \quad (\text{B.33})$$

We can then substitute this expression back into eq. (B.29) resulting in

$$\mathcal{L}_{\text{cross-entropy}} = \int q_{\epsilon}(\boldsymbol{\epsilon}^{(r)}) \log \frac{q_{\epsilon}(g_{\text{prior}}^{-1}(g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})))}{\left| (\epsilon_0^{(r)})^{D-1} \prod_{i=2}^D (\sin(\epsilon_i^{(r)}))^{i-1} \prod_j \sigma_{\text{prior},j} \right|}. \quad (\text{B.34})$$

Because the denominator is independent of the optimized variables, we can neglect the constant and focus on

$$= \int q_{\epsilon}(\boldsymbol{\epsilon}^{(r)}) \log q_{\epsilon}(g_{\text{prior}}^{-1}(g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)}))), \quad (\text{B.35})$$

where we can use the definition of the change of variables to see

$$= \int q_{\epsilon}(\boldsymbol{\epsilon}^{(r)}) \log q_{\epsilon} \left(\mathbf{T}_{xr} \left(\frac{\mathbf{w}^{(x)} - \boldsymbol{\mu}_{\text{prior}}}{\boldsymbol{\sigma}_{\text{prior}}} \right) \right). \quad (\text{B.36})$$

That is, we can compute the expectation by finding the hyperspherical coordinate-system density of the transformed normalized weights. The transformation is standard and widely used, but computationally inconvenient, making it much more efficient to use a Gaussian prior where possible. We can estimate it using a Monte Carlo approximation:

$$\approx \frac{1}{N} \sum_{i=1}^N \log q_{\epsilon} \left(\mathbf{T}_{xr} \left(\frac{\mathbf{w}^{(x)} - \boldsymbol{\mu}_{\text{prior}}}{\boldsymbol{\sigma}_{\text{prior}}} \right) \right). \quad (\text{B.37})$$

Recalling eq. (B.19) which states the probability density function of the Radial distribution in hyperspherical coordinates

$$q(\boldsymbol{\epsilon}^{(r)}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}} \cdot \prod_{i=1}^{D-1} \sin(\epsilon_i^{(r)})^{D-i}, \quad (\text{B.38})$$

we can then compute the cross-entropy term of the loss using an explicit coordinate transformation and this expression. This is much more computationally cumbersome than the multivariate Gaussian case, but is tractable.

B.3 Experimental Settings

Diabetic Retinopathy Settings

Authors' Note: The version of the diabetic retinopathy dataset used in this paper (as described below) is slightly different from the one used in the latest release of the Diabetic Retinopathy benchmark (as of May 2021). Radial BNNs have been included in a more recent benchmarking effort in which they do not perform quite as well as we found originally.

The diabetic retinopathy data are publicly available at <https://www.kaggle.com/c/diabetic-retinopathy-detection/data>. We augment and preprocess them similarly to Leibig et al. [2017]. The images for our main experiments in section 4.3 are downsampled to 512x512 while the smaller robustness experiment in section 4.3 uses images downsampled to 256x256. We randomly flip horizontally and vertically. Then randomly rotate 180 degrees in either direction. Then we pad by between 0 and 5% of the width and height and randomly crop back down to the intended size. We then randomly crop to between 90% and 110% of the image size, padding with zeros if needed. We finally resize again to the intended size and normalize the means and standard deviations of each channel separately based on the training set means and standard deviations. The training set has 44,594 RGB images. There are 7,026 validation and 10,000 test images.

The smaller model used for robustness experiments is loosely inspired by VGG-16, with only 16 channels, except that it is a Bayesian neural network with mean and standard deviations for each weight, and that instead of fully connected networks at the end it uses a concatenated global mean and average pool. The larger model used in the main experiments is VGG-16 but with the concatenated global mean and average pool instead of fully connected layers as above. The only difference is that we use only 46 channels, rather than 64 channels as in VGG-16, because the BNN has twice as many parameters as a similarly sized deterministic network, and we wanted to compare models with the same number of parameters. For the dropout model we use VGG-16 with the full 64 channels, and similarly for each of the models in the deep ensemble. The prior for training MFVI and Radial BNNs was a unit multivariate Gaussian. (We also tried using the scale mixture prior used in Blundell et al. [2015] and found it made no difference.) Instead of optimizing σ directly we in fact optimize ρ such that $\sigma = \log(1+e^\rho)$ which guarantees that σ is always positive. In some cases, as described in the paper, the first epoch only trained the means and uses a NLL loss function. This helps the optimization, but in principle can still allow the variances to train fully if early stopping is not employed (unlike reweighting the KL-divergence). Thereafter, we trained using the full ELBO loss over all parameters. Unlike some prior work using MFVI, we have not downweighted the KL-divergence during training.

For the larger models, we searched for hyperparameters using Bayesian optimization. We searched between 0 and -10 as the initial value of ρ (equivalent to σ values of $\log(2)$ and $2 \cdot 10^{-9}$). For the learning rate we considered 10^{-3} to 10^{-5} using Adam with a batch size of 16. Otherwise, hyperparameters we based on exploration from the smaller model.

We then computed the test scores using a Monte Carlo estimate from averaging 16 samples from the variational distribution. We estimate the model’s uncertainty about a datapoint using the mutual information between the posterior’s parameters and the prediction on a datapoint. This estimate is used to rank the datapoints in order of confidence and compute the model’s accu-

racy under the assumption of referring increasingly many points to medical experts.

For the smaller models, we performed an extensive random hyperparameter search. We tested each configuration with both MFVI and Radial BNNs. We tested each configuration for both an SGD optimizer and Amsgrad. When training with SGD we used Nesterov momentum 0.9 and uniformly sampled from 0.01, 0.001 and 0.0001 as learning rates, with a learning rate decay each epoch of either 1.0 (no decay), 0.98 or 0.96. When training with Amsgrad we uniformly sampled from learning rates of 0.001, 0.0001, and 0.00001 and did not use decay. We uniformly selected batch sizes from 16, 32, 64, 128, and 256. We uniformly selected the number of variational distribution samples used to estimate the loss from 1, 2, and 4. However, because we discarded all runs where there was insufficient graphics memory, we were only able to test up to 64x4 or 256x1 and batch sizes above 64 were proportionately less likely to appear in the final results. We selected the initial variance from ρ values of -6, -4, -2, or 0. We also tried reducing the number of convolutional channels by a factor of 5/8 or 3/8 and found that this did not seem to improve performance. We ran our hyperparameter search runs for 150 epochs. We selected the best hyperparameter configurations based on the best validation accuracy at any point during the training. We trained the models for 500 epochs but selected the models saved from 300 epochs as all models had started to overfit by the end of training. For MFVI, this was using the SGD optimizer with learning rate 0.001, decay rate 0.98 every epoch, batch size 16, 4 variational samples for estimating the loss during training and ρ of -6. This outperformed the others by a significant margin. Using our code on a V100 GPU with 8 vCPUs and an SSD this took slightly over 13 hours to train each model. For the radial posterior, this was the Adam optimizer with learning rate 0.0001, batch size 64, 1 variational sample for estimating the loss during training and a ρ of -6. Using our code on the same GPU, this took slightly over 3h to run. However, for the radial posterior there were very many other configurations with similar validation accuracies (one of the advantages of the posterior).

For the experiment shown in Figure 4.7, we have selected slightly different hyperparameters in order to train more quickly. For both models, we use Adam with learning rate 0.0001 and train for 500 epochs. The models have 5/8 the number of channels of VGG-16. The models are trained with batch size 64 and 4 variational samples to estimate the loss and its standard deviation.

Appendix C

Appendix to Evaluating Bayesian Deep Learning

C.1 Experimental Details

Linear Regression

Our training dataset contains a small cluster of points near $x = -1$ and two larger clusters at $0 \leq x \leq 0.5$ and $1 \leq x \leq 1.5$, sampled proportionately to the ‘true’ data distribution. The data distribution from which we select data in a Rao-Blackwellised manner has a probability density function over x equal

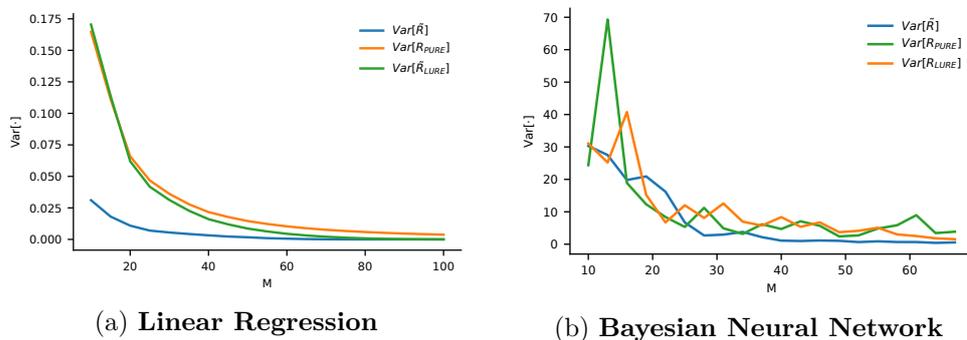


Figure C.1: For linear regression (a) the biased estimator has the lowest variance, and \tilde{R}_{LURE} improves on \tilde{R}_{PURE} . (b) But for the BNN the variances are more comparable, with \tilde{R}_{LURE} the lowest.

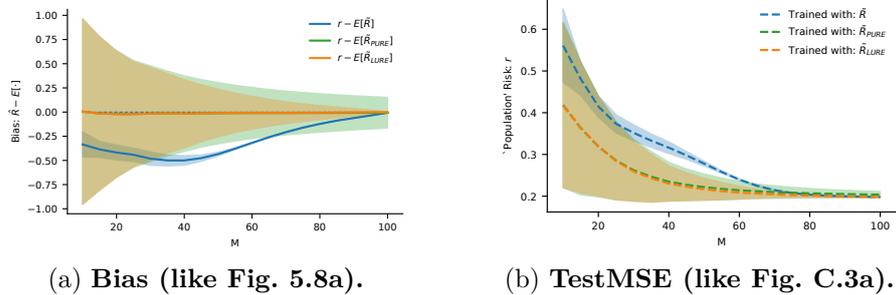


Figure C.2: Adopting an alternative proposal distribution—here an epsilon-greedy adaptation of a distance-based measure—does not change the overall picture for linear regression.

to:

$$P(x = X) = \begin{cases} 0.12 & -1.2 \leq x \leq -0.8 \\ 0.95 & 0.0 \leq x \leq 0.5 \\ 0.95 & 1.0 \leq x \leq 1.5 \end{cases} \quad (\text{C.1})$$

while the distribution over y is then induced by:

$$y = \max(0, x) \cdot \left(|x|^{\frac{3}{2}} + \frac{\sin(20x)}{4} \right). \quad (\text{C.2})$$

We set $N = 101$, where there are 5 points in the small cluster and 96 points in each of the other two clusters, and consider $10 \leq M \leq 100$. We actively sample points without replacement using a geometric heuristic that scores the quadratic distance to previously sampled points and then selects points based on a Boltzman distribution with $\beta = 1$ using the normalized scores.

Here, we also show in Figure C.2 results that are collected using an epsilon-greedy acquisition proposal. The results are aligned with those from the other acquisition distribution we consider in the main body of the paper. This proposal selects the point that is has the highest total distance to all previously selected points with probability 0.9 and uniformly at random with probability $\epsilon = 0.1$. That is, the acquisition proposal is given by:

$$P(i_m = j; i_{1:m-1}, \mathcal{D}_{\text{pool}}) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{D}_{\text{pool}}|} & \arg \max_{j \notin \mathcal{D}_{\text{train}}} \sum_{k \in \mathcal{D}_{\text{train}}} |x_k - x_j| \\ \frac{\epsilon}{|\mathcal{D}_{\text{pool}}|} & \text{otherwise} \end{cases} \quad (\text{C.3})$$

where of course $\mathcal{D}_{\text{train}}$ are the $i_{1:m-1}$ elements of $\mathcal{D}_{\text{pool}}$.

For all graphs we use 1000 trajectories with different random seeds to calculate error bars. Although, of course, each regression and scoring is deterministic, the acquisition distribution is stochastic.

Although the variance of the estimators can be inferred from Figure 5.8a, we also provide Figure C.1a which displays the variance of the estimator directly.

Hyperparameter	Setting description
Architecture	Convolutional Neural Network
Conv 1	1-16 channels, 5x5 kernel, 2x2 max pool
Conv 2	16-32 channels, 5x5 kernel, 2x2 max pool
Fully connected 1	128 hidden units
Fully connected 2	10 hidden units
Loss function	Negative log-likelihood
Activation	ReLU
Approximate Inference Algorithm	Radial BNN Variational Inference [Farquhar et al., 2020]
Optimization algorithm	Amsgrad [Reddi et al., 2018]
Learning rate	$5 \cdot 10^{-4}$
Batch size	64
Variational training samples	8
Variational test samples	8
Variational acquisition samples	100
Epochs per acquisition	up to 100 (early stopping patience=20), with 1000 copies of data
Starting points	10
Points per acquisition	1
Acquisition proposal distribution	$q(i_m; i_{1:m-1}, \mathcal{D}_{\text{pool}}) = \frac{e^{T s_i}}{\sum e^{T s_i}}$
Temperature: T	10,000
Scoring scheme: s	BALD (M.I. between θ and output distribution)
Variational Posterior Initial Mean	He et al. [2016]
Variational Posterior Initial Standard Deviation	$\log[1 + e^{-4}]$
Prior	$\mathcal{N}(0, 0.25^2)$
Dataset	Unbalanced MNIST
Preprocessing	Normalized mean and std of inputs.
Validation Split	1000 train points for validation
Runtime per result	2-4h
Computing Infrastructure	Nvidia RTX 2080 Ti

Table C.1: Experimental Setting—Active MNIST.

Bayesian Neural Network

We train a Bayesian neural network using variational inference [Jordan et al., 1999]. In particular, we use the radial Bayesian neural network approximating distribution [Farquhar et al., 2020]. The details of the hyperparameters used for training are provided in Table C.1.

The unbalanced dataset is constructed by first noising 10% of the training labels, which are assigned random labels, and then selecting a subset of the training dataset such that the numbers of examples of each class is proportional to the ratio (1., 0.5, 0.5, 0.2, 0.2, 0.2, 0.1, 0.1, 0.01, 0.01)—that is, there are 100 times as many zeros as nines in the unbalanced dataset. (Figure C.3l shows a version of this experiment which uses a balanced dataset instead, in order to make sure that any effects are not entirely caused by this design choice.) In fact, we took only a quarter of this dataset in order to speed up acquisition (since each model must be evaluated many times on each of the candidate datapoints to estimate the mutual information). 1000 validation points were then removed from this pool to allow early stopping. The remaining points were placed in $\mathcal{D}_{\text{pool}}$. We then uniformly selected 10 points from $\mathcal{D}_{\text{pool}}$ to place in $\mathcal{D}_{\text{train}}$. Adding noise to the labels and using an unbalanced dataset is designed to mimic the difficult situations that active learning systems are deployed on in practice, despite the relatively simple

dataset. However, we used a simple dataset for a number of reasons. Active learning is very costly because it requires constant retraining, and accurately measuring the properties of estimators generally requires taking large numbers of samples. The combination makes using more complicated datasets expensive. In addition, because our work establishes a lower bound on architecture complexity for which correcting the active learning bias is no longer valuable, establishing that lower bound with MNIST is in fact a stronger result than showing a similar result with a more complex model.

The active learning loop then proceeds by:

1. training the neural network on $\mathcal{D}_{\text{train}}$ using \tilde{R} ;
2. scoring $\mathcal{D}_{\text{pool}}$;
3. sampling a point to be added to $\mathcal{D}_{\text{train}}$;
4. Every 3 points, we separately trained models on $\mathcal{D}_{\text{train}}$ using \tilde{R} , \tilde{R}_{PURE} , and \tilde{R}_{LURE} and evaluate them.

This ensures that all of the estimators are on data collected under the same sampling distribution for fair comparison. As a sense-check, in Figures C.4a and C.4b we show an alternate version in which the first step trains with \tilde{R}_{LURE} instead of \tilde{R} , and find that this does not have a significant effect on the results.

When we compute the bias of a fixed neural network in Figure 5.8b, we train a single neural network on 1000 points. We then sample evaluation points using the acquisition proposal distribution from the test dataset and evaluate the bias using those points.

In Figures C.5a and C.5b we review the graphs shown in Figures C.3b and C.3c, this time showing standard errors in order to make clear that the biased \tilde{R} estimator has better performance, while the earlier figures show that the performance is quite variable.

We considered a range of alternative proposal distributions. In addition to the Boltzman distribution which we used, we considered a temperature range between 1,000 and 20,000 finding it had relatively little effect. Higher temperatures correspond to more certainly picking the highest mutual information point, which approaches a deterministic proposal. We found that because the mutual information had to be estimated, and was itself a random variable, different trajectories still picked very different sets of points. However, for very high temperatures the estimators became higher variance, and for lower temperatures, the acquisition distribution became nearly uniform. In Figure C.6 we show the results of networks trained with a variety of temperatures other than the 10,000 ultimately used. We also considered a proposal which was simply proportional to the scores, but found this was also too close to sampling uniformly for any of the costs or benefits of active learning to be visible.

Reference	Application	Corrects Bias	Acknowledges Bias	Notes
[Sener and Savarese, 2018]				
[Shen et al., 2018]	✓			
[Beluch et al., 2018]				
[Haut et al., 2018]	✓			
[Sinha et al., 2019]				
[Siddhant and Lipton, 2018]	✓		✓	
[Ghosal et al., 2019]	✓			
[Yang et al., 2018]	✓			
[Yoo and Kweon, 2019]				
[Kirsch et al., 2019]				
[Huang et al., 2018]			✓	
[Wen et al., 2018a]	✓			
[Chen et al., 2019]	✓			Discusses bias in $\mathcal{D}_{\text{pool}}$.
[Zhang and Lee, 2019]	✓			Discusses bias in $\mathcal{D}_{\text{pool}}$.
[Kellenberger et al., 2019]	✓			

Table C.2: Existing applications of deep active learning rarely acknowledge the bias introduced by actively sampling points and do not, to the best of our knowledge, try to correct it.

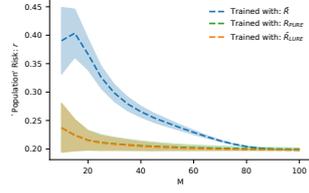
We considered Monte Carlo dropout as an alternative approximating distribution [Gal and Ghahramani, 2015] (see Figures C.3h and C.7b). We found that the mutual information estimates were compressed in a fairly narrow range, consistent with the observation by Osband et al. [2018] that Monte Carlo dropout uncertainties do not necessarily converge unless the dropout probabilities are also optimized [Gal et al., 2017a]. While this might be good enough when only the *relative* score is needed in order to calculate the argmax, for our proposal distribution we would ideally prefer to have good *absolute* scores as well. For this reason, we chose the richer approximate posterior distribution instead.

Last, we considered a different architecture, using a full-connected neural network with a single hidden layer with 50 units, also trained as a Radial BNN. This showed higher variance in downstream performance, but was broadly similar to the convolutional architecture (see Figures C.7d and C.7e).

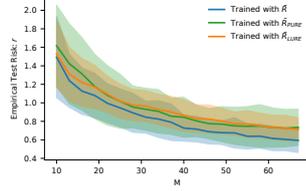
C.2 Deep Active Learning In Practice

In Table C.2, we show an informal survey of highly cited papers citing Gal et al. [2017b], which introduced active learning to computer vision using deep convolutional neural networks. Across a range of papers including theory papers as well as applications ranging from agriculture to molecular science only two papers acknowledged the bias introduced by actively sampling and none of the papers took steps to address it. It is worth noting, though, that at least two papers motivated their use of active learning by observing that they expected their training data to already be unrepresentative of the population data and saw active learning as a way to address *that* bias. This does not

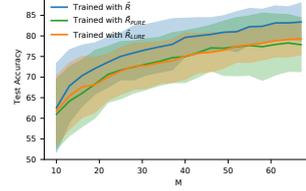
quite work, unless you explicitly assume that the actively chosen distribution is more like the population distribution, but is an interesting phenomenon to observe in practical applications of active learning.



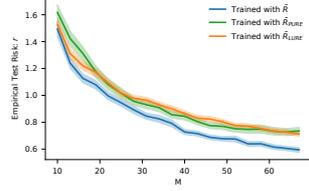
(a) **Linear regression.**
Test MSE.



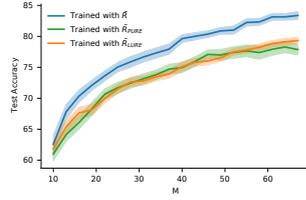
(b) **MNIST: Test NLL.**



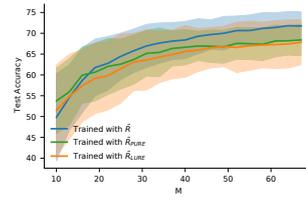
(c) **MNIST: Test Acc.**



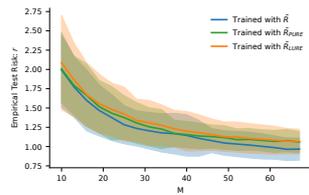
(d) **MNIST: Test NLL.**
Standard Error



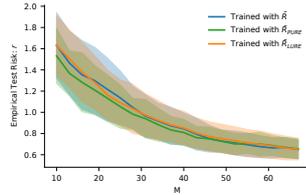
(e) **MNIST: Test Acc.**
Standard Error



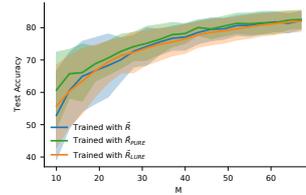
(f) **FashionMNIST: Test Acc.**



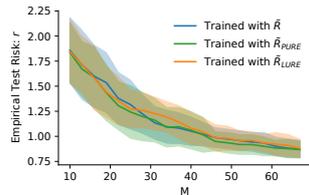
(g) **FashionMNIST:**
Test NLL.



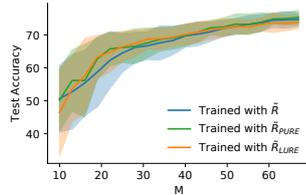
(h) **MNIST (MCDO):**
Test NLL.



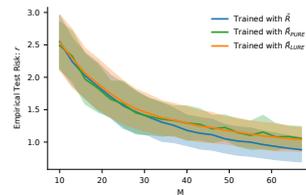
(i) **MNIST (MCDO):**
Test Acc.



(j) **FashionMNIST**
(MCDO): Test NLL.



(k) **FashionMNIST**
(MCDO): Test Acc.



(l) **MNIST (Balanced):**
Test NLL.

Figure C.3: For linear regression, the models trained with \tilde{R}_{PURE} or \tilde{R}_{LURE} have lower ‘population’ risk. In contrast, BNNs trained with \tilde{R}_{PURE} or \tilde{R}_{LURE} perform either similarly (h, i) or slightly worse (b-g), even though they remove bias and have lower variance. Shading is one standard deviation to show variation, except for (d-e) where we show standard error to demonstrate the significance of the differences. For (a) 1000 samples and ‘ r ’ estimated on 10,100 points from distribution. Otherwise 45 samples and ‘ r ’ estimated on the test dataset.

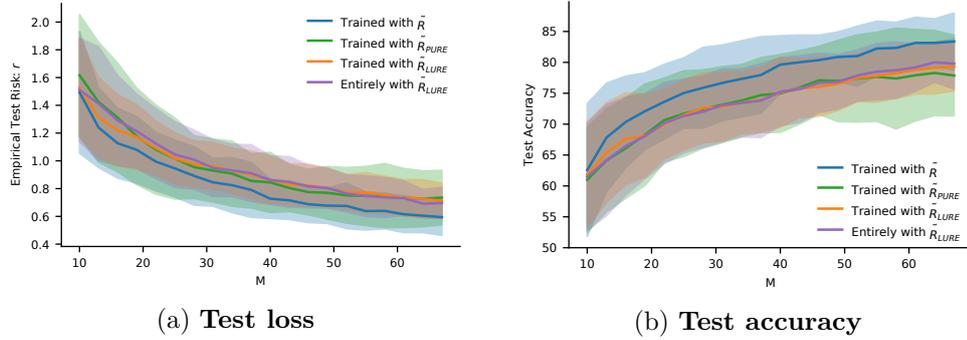


Figure C.4: We contrast the effect of using \tilde{R}_{LURE} throughout the entire acquisition procedure and training (rather than using the same acquisition procedure based on \tilde{R} for all estimators). The purple test performance and orange are nearly identical, suggesting the result is not sensitive to this choice.

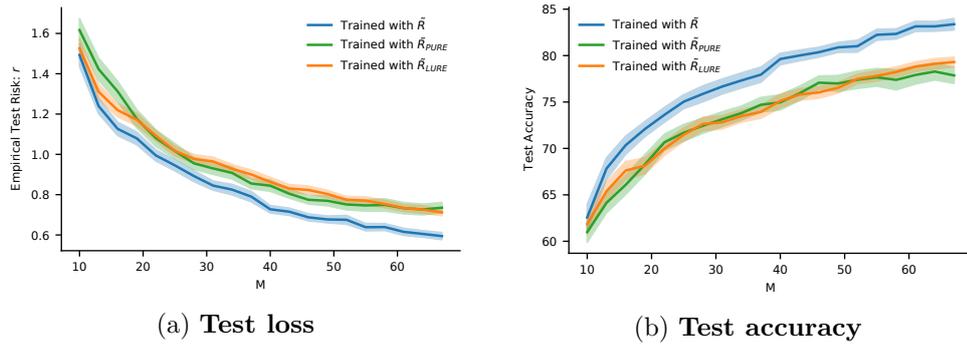


Figure C.5: Versions of Figures C.3b and C.3c shown with standard errors (45 points) instead of standard deviations. This makes it clearer that the biased \tilde{R} has better performance, even if only marginally so.

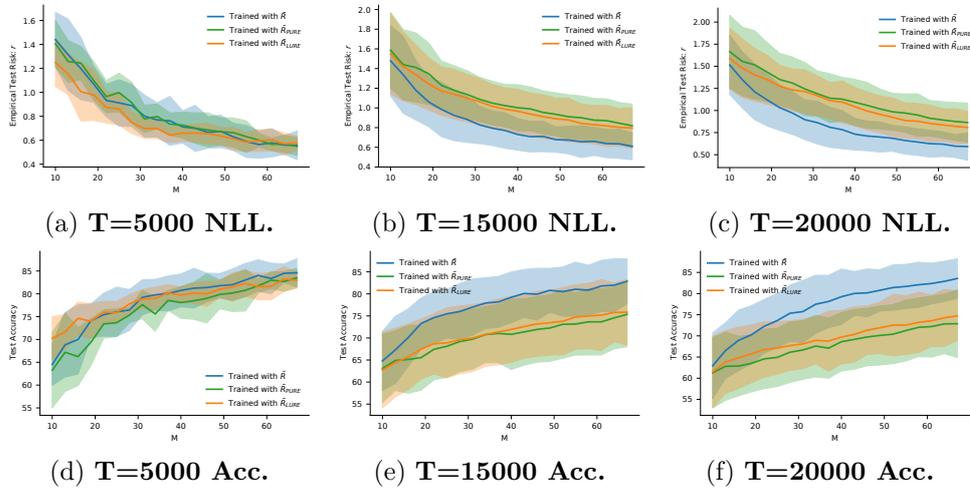


Figure C.6: Higher temperatures approach a deterministic acquisition function. These also tend to increase the variance of the risk estimator because the weight associated with unlikely points increases, when it happens to be selected. The overall pattern seems fairly consistent, however.

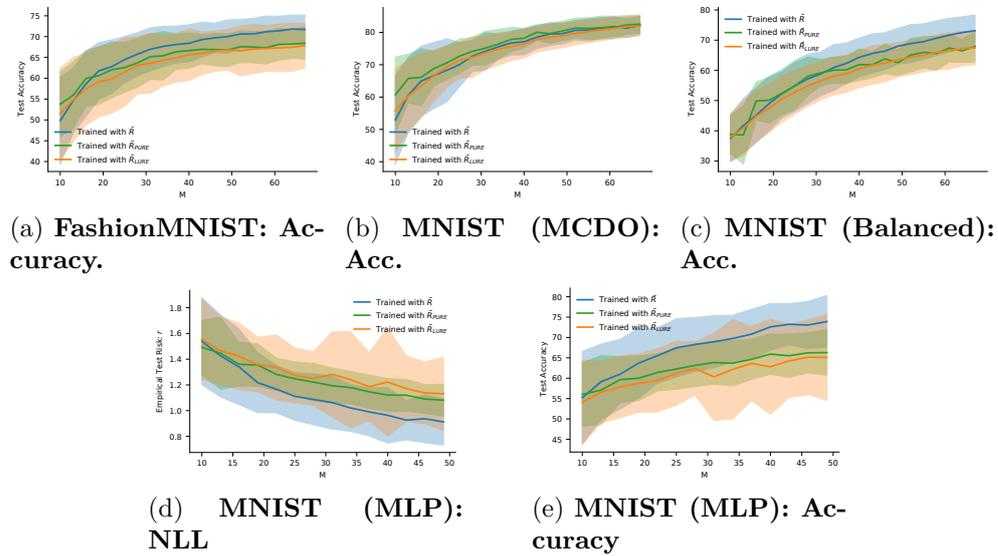


Figure C.7: Further downstream performance experiments. (a)-(c) are partners to Figures C.3g, C.3h, and C.3l. (d) and (e) show similar results for a smaller multi-layer perceptron (with one hidden layer of 50 units). In all cases the results broadly mirror the results in the main paper.