Local vs. Global Uncertainty in Bayesian Deep Learning

by

Thomas G. Grigg

supervised by

Hippolyt Ritter & David Barber

Submitted as part requirement for the degree of *Master of Science in Computational Statistics and Machine Learning* at University College London[1].

September 2020
Centre for CSML
University College London

[1]This report is substantially the result of my own work except where explicitly indicated in the text. This report may be freely copied and distributed provided the source is explicitly acknowledged.

Acknowledgements

This project would not have been possible without the consistent support and guidance of Hippolyt Ritter, without whom I myself would have been quite uncertain, both locally and globally. I am also greatly appreciative of David for his kind agreement to support and supervise this project.

Abstract

Deep ensembles have emerged as the established method for representing parametric uncertainty in deep neural networks [38]. A recent practical alternative known as *Stochastic Weight Averaging-Gaussian* instead fits the iterates of stochastic gradient descent to a Gaussian distribution with a computationally efficient covariance structure in order to model the distribution of weights around a single mode [43]. The work contained within these pages is primarily an empirical investigation into the nature of the two approaches in practice, with particular focus given to exploring the trade-offs associated with representing uncertainty across the parameters corresponding to different modes (i.e. “global” uncertainty), and representing uncertainty within a given mode via a SWA-Gaussian approximation (i.e. "local" uncertainty). We demonstrate that local uncertainty primarily benefits model calibration, whereas representing global uncertainty more strongly benefits predictive performance. Furthermore, we find that in the natural combination of these two methods, the benefit of each type of uncertainty is not easily obtained simultaneously. We additionally make clear the storage and computation trade-offs with respect to the empirical performances of deep ensembles, SWA-Gaussian, and a number of their variants, culminating in simple practical recommendations for practitioners wishing to make use of the methods for uncertainty representation. Finally, we challenge some common assumptions about diversity in the context of classification ensembles, as well as the popular idea that ensembling necessarily leads to well-calibrated models.

Contents

Acknowledgements i

Abstract ii

1 Introduction 2

1.1 Uncertainty in Deep Neural Networks . . . 2

1.1.1 Global Uncertainty . . . 3

1.1.2 Local Uncertainty . . . 5

1.2 Objectives and Contributions . . . 7

2 Background and Related Work 8

2.1 Modelling Uncertainty . . . 8

2.1.1 Why Model Predictive Uncertainty in Deep Learning? . . . 9

2.1.2 Aleatoric and Epistemic Uncertainty . . . 10

2.2 Probabilistic Supervised Learning . . . 11

2.2.1 Empirical Risk Minimisation in the Probabilistic Setting . . . 11

2.2.2 Learning Distributions with Neural Networks . . . 13

2.2.3 Issues with Overconfidence in ERM . . . 15

2.3 Bayesian Deep Learning . . . 17

2.3.1 Bayesian Model Averaging . . . 18

2.3.2 Variational Inference . . . 20

2.3.3 Markov Chain Monte Carlo . . . 21

2.3.4 SGD as Approximate Bayesian Inference . . . 22

2.4 Local and Global Methods for Uncertainty . . . 23

2.4.1 Stochastic Weight Averaging-Gaussian (SWAG) . . . 23

2.4.2 Deep Ensembles . . . 27

2.4.3 MultiSWAG . . . 30

3 Experiments and Results **31**
3.0.1 Method Complexities . . . 31
3.0.2 Architectures and Datasets . . . 32
3.0.3 Metrics . . . 33
3.1 Performance as Individual Solutions . . . 34
3.2 Comparison as Ensembling Techniques . . . 36
3.2.1 Underconfidence of Deep Ensembles . . . 38
3.3 Analysis of SWA-Gaussian's Local Subspace . . . 39
3.3.1 Performance versus Subspace Rank . . . 39
3.3.2 Covariance and Scale . . . 41
3.3.3 How Important is the Diagonal? . . . 43
3.4 Analysis of Diversity . . . 45
3.4.1 Pairwise Diversity of Methods . . . 45
3.4.2 Local versus Global Diversity . . . 48
3.5 Representing Both Local and Global Uncertainty . . . 49

4 Conclusion **55**
4.1 Discussion . . . 55
4.2 Future Work . . . 57

References **58**

Notation

We will try to be clear about notation inline, but present our conventions:

$\mathcal{X}$	A set of inputs or observations; also a random variable taking values in this set.
$\mathcal{Y}$	A set of outputs, responses, or assignments; also a random variable taking values in this set.
$Distr(\mathcal{Y})$	The set of possible distributions/probability measures over $\mathcal{Y}$. Usually constrained to be masses or densities.
$\mathcal{A}$	A neural network architecture, usually considered as a mapping $\mathcal{A} : \Theta \to (\mathcal{X} \to \mathcal{Y})$ for a predictive model and $\mathcal{A} : \Theta \to (\mathcal{X} \to Distr(Y))$ for a probabilistic model.
Θ	A space of parameters. $\Theta \subseteq \mathbb{R}^{\lvert\mathcal{A}\rvert}$ where $\lvert\mathcal{A}\rvert$ denotes the number of weight parameters of an architecture.
θ	An arbitrary setting of parameters $\theta \in \Theta$.
$\mathcal{H}_{\mathcal{A}}$	The space of all possible hypotheses representable by an architecture. $\mathcal{H}_{\mathcal{A}} = \{\mathcal{A}(\theta) \lvert \theta \in \Theta\}$.
f	A predictive model $f : \mathcal{X} \to \mathcal{Y}$.
p	A model of a predictive probability distribution $p : \mathcal{X} \to Distr(\mathcal{Y})$.
p_θ	Shorthand for $\mathcal{A}(\theta)$, an element of $\mathcal{H}_{\mathcal{A}}$.
$\mathbb{P}(\mathcal{Y})$	Marginal probability distribution over $\mathcal{Y}$.
$\mathbb{P}(\mathcal{Y}\lvert\mathcal{X})$	Conditional probability distribution of $\mathcal{Y}$ given $\mathcal{X}$,
$\mathbb{P}(y)$ or $\mathbb{P}(\mathcal{Y} = y)$	Probability mass or density of $y \in \mathcal{Y}$ (similarly for conditionals).
$p(x)$	Probabilistic model evaluated at $x \in \mathcal{X}$, i.e. $p(x) \in Distr(\mathcal{Y})$
$p(x)(y)$ or $p(y\lvert x)$	Mass/density function corresponding to probabilistic model evaluated at $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Finally, note that we use Lebesgue notation unless we want to be explicit about masses or densities, e.g. $\mathbb{P}(\mathcal{Y}) = \int_{x \in \mathcal{X}} \mathbb{P}(\mathcal{Y}, \mathcal{X} = x) d\mathbb{P}(\mathcal{X} = x)$ or just for shorthand $\mathbb{P}(\mathcal{Y}) = \int_{\mathcal{X}} \mathbb{P}(\mathcal{Y}, \mathcal{X}) d\mathbb{P}(\mathcal{X})$.

Chapter 1

Introduction

It is well established that deep neural networks (DNNs) are extraordinarily powerful black-box models that can be used for a wide class of statistical learning problems to great effect. In particular, due to their flexibility and ability to learn useful internal representations essentially automatically, they have a relatively long history as predictive models. Since at least the early 1990s [39], DNNs have played a significant role in pushing forward the state-of-the-art on an increasingly diverse range of predictive tasks, even famously surpassing human experts in certain narrow domains in recent years [55, 16]. With the advent of the 2020s, DNNs are deployed within decision-making processes throughout industry, and their prevalence is likely to continue to increase over the coming years as artificially intelligent technology becomes more and more commonplace in our society.

A question of growing importance is therefore: *how much can we trust the predictions of a given deep neural network?* If a predictive model is deployed as part of an autonomous decision making system, such as a self-driving car or medical diagnostic tool [16, 63], then even a single incorrect prediction could have extremely serious consequences. One of the primary motivations for endowing DNNs with explicit predictive uncertainty estimates is thus to help us to identify and mitigate against unreliable predictions.

1.1 Uncertainty in Deep Neural Networks

This project is concerned with the representation of uncertainty in DNNs applied to predictive modelling. Idealistically, our goal is to explore the concepts of "local" and "global" uncertainty, which we shall soon make clear. Pragmatically, we explore and compare two state-of-the-art methods for representing uncertainty in DNNs, each method embodying one of these notions. In this first chapter, we clarify the concepts of "local" and "global" uncertainty, and briefly discuss their respective methods SWA-Gaussian and Deep Ensembles, ultimately motivating our empirical investigation into the two notions/methods and their interactions.

1.1.1 Global Uncertainty

Recall that the parameters of a neural network are usually optimised via some variant of stochastic gradient descent, whereby we iteratively attempt to move in a loss-decreasing direction. With good choices for hyperparameters, this procedure works remarkably well: we invariably arrive at a strong local solution. In fact, empirically we tend to arrive at an effectively global minimum, i.e. achieving essentially optimal training loss [14]. Traditionally, after we arrive at such a minimum and we have confirmed that it has good generalisation to unseen data via validation, the minimum's corresponding parameters are taken to define the final model. Many state-of-the-art models in numerous domains are some form of deep neural network trained in this way. However, neural networks typically have millions of parameters, and thus are often underspecified by the labelled datasets that they are applied to, which themselves may only have thousands of datapoints. In other words, many different settings of a neural network's parameters often correspond to strong models for the data. Given that neural network training can be interpreted geometrically as a (stochastic) descent over a loss surface induced by some target loss $\mathcal{L}$ with respect to a given set of labelled examples $\mathcal{D}$, this can be rephrased geometrically as: neural network loss surfaces are typically non-convex and have many *modes*[1], i.e. they are *multimodal*.

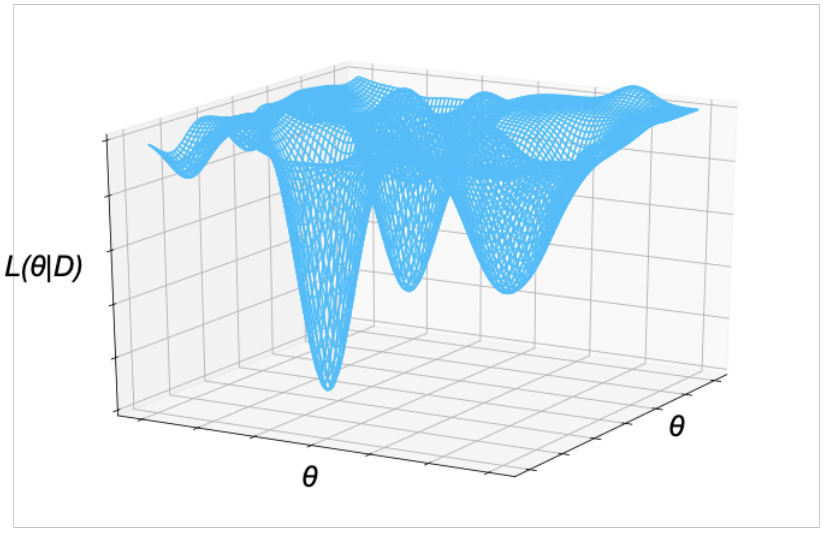
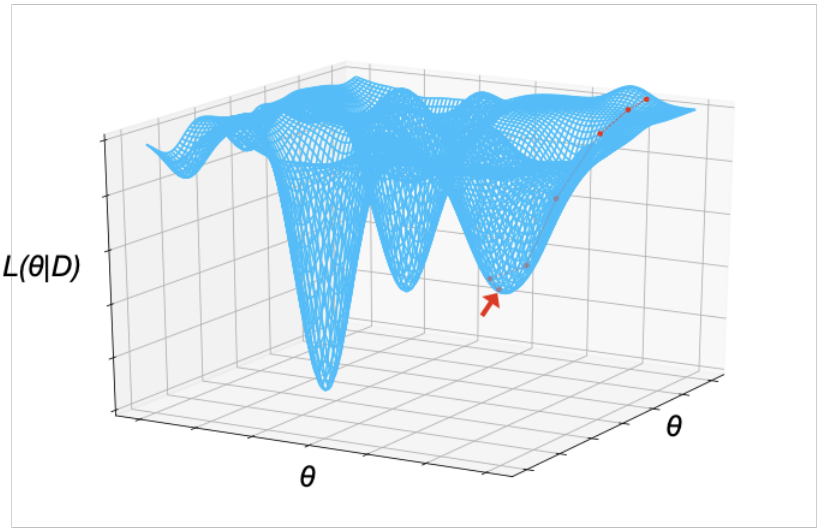


Figure 1.1: An illustration of a θ-parameterised neural network loss surface with three distinct *modes*.

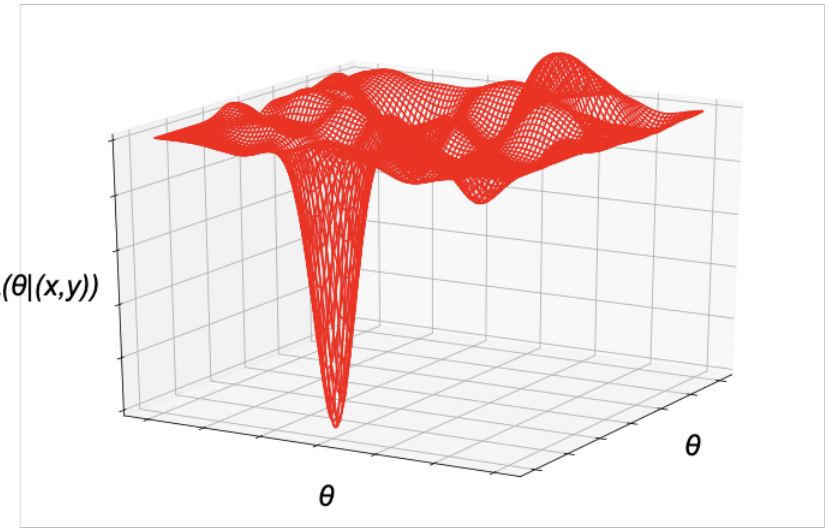
[1]Note that we use the term *mode* loosely here (and throughout) to refer to local "roughly" convex regions of low loss in parameter space [42] - another term found in the literature for this is "basin of attraction". In this sense, a mode contains many (local) minima.

Due to this multimodality, a given minimum arrived at by gradient descent is unlikely to be unique: it is likely that the loss surface admits entirely different minima that are equally as strong (or perhaps stronger). In fact, running exactly the same optimization method with identical hyperparameters[2] but a different initialisation is often all it takes to arrive at an entirely different minimum corresponding to parameters with similar performance. Two distinct parameterisations arrived at in this manner almost certainly correspond to two different functions [17], and issues with picking one over the other arise when these functions disagree significantly.

For an example, let us suppose that our optimisation brings us to the 'rightmost' minimum on our idealised training loss surface:



And that after training we are then given a previously unseen datapoint (x, y) whose loss surface looks like:



[2]This is in contrast to most other models (particularly those with low training variance), which require *bootstrapping* to be effectively ensembled.

In this contrived example we see that the parameters selected via the original optimization perform poorly with respect to L on the unseen point (x, y). However, if we had instead arrived at parameters corresponding to the 'leftmost' mode, we would have performed well on (x, y). This situation exemplifies the importance of our *global* uncertainty in the model's parameters: we want to capture the variability around which mode we ultimately end up in given different arbitrary parameter initialisations. Intuitively, we might posit that the different modes of a neural network correspond to parameter settings which encode potentially different but equally convincing "explanations" for the observed data.

Deep Ensembles

Deep ensembles [38] are the established method for dealing with this global notion of parametric uncertainty. In a deep ensemble, we form a more robust solution by mixing the predictions of models corresponding to multiple minima, each within a distinct mode. Not only does this tend to produce better predictions by mitigating against situations similar to the above example, the resulting mixture model also typically gives us a better representation of model uncertainty. We provide a more comprehensive introduction to deep ensembles in Section 2.4.2.

1.1.2 Local Uncertainty

While global uncertainty captures uncertainty related to the global position/specific mode which we end up in after our descent over the fixed training loss surface, *local* uncertainty focuses on the more subtle variability in the local shape of the training loss surface itself with respect to the observed data.

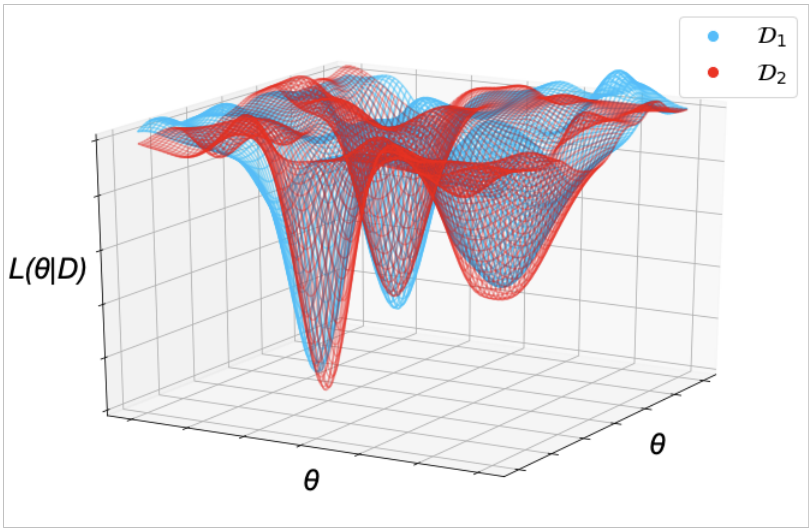


Figure 1.2: An illustration of the neural network loss surfaces for two different sampled datasets.

Supposing as before we optimise and arrive somewhere in the "rightmost" mode of the training dataset (blue). How well does this setting of parameters perform on the unseen data (red)?

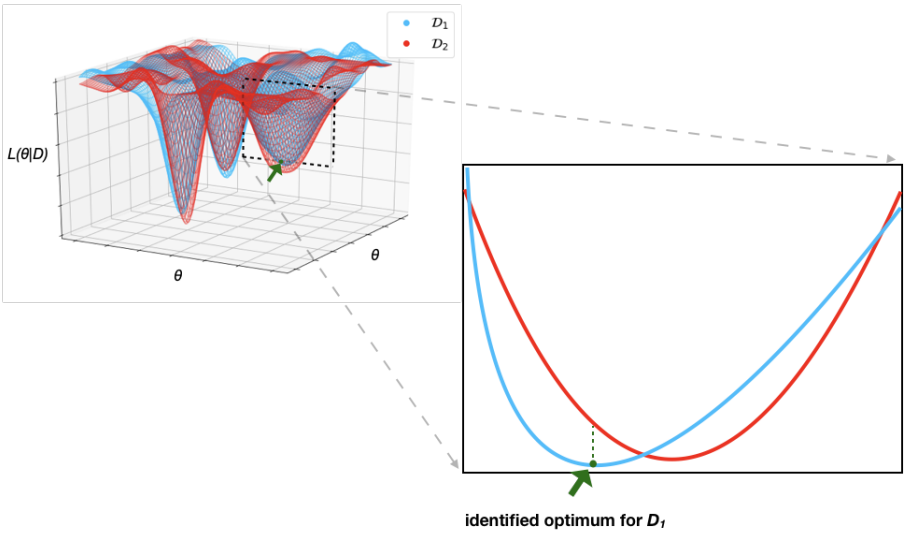


Figure 1.3: An illustration depicting a slice into the (idealised) neural network loss surfaces for two different sampled datasets. Observe that the learned optimum identified for dataset $\mathcal{D}_1$ is not quite optimal for dataset $\mathcal{D}_2$.

As we can see in Figure 1.3, predicting with the parameters corresponding to a local optimum for one sampled dataset may no longer be locally optimal with respect to another sampled dataset due to natural variance in the sampling process. This captures the idea behind *local* uncertainty: the precise location for the optimal setting of parameters within a given mode varies with the training data. In this case we might intuit that the "fine details" of each "explanation" for the observed data differ depending on the observed examples.

Bayesian Deep Learning and SWA-Gaussian

Traditionally, Bayesian deep learning is the subfield of deep learning focused on handling this local form of parametric uncertainty. The general idea is to place a prior probability distribution over the model parameters, and to perform Bayesian inference using the observed data to compute a posterior distribution. Due to the high-dimensionality of a neural network's parameter space, exact inference is usually not possible and so a variety of methods exist to approximate the posterior, typically focusing on the variation within a single mode. Stochastic Weight Averaging-Gaussian (SWA-Gaussian or SWAG) [43] is one such method that has recently been shown to be practical and effective. SWA-Gaussian performs stochastic gradient descent with a decreasing learning rate in order to

converge to a local optimum as normal, switching to a fixed constant learning rate after convergence in order to explore the local region around the identified optimum. While doing so, the method uses the parameter samples it collects to construct a Gaussian local approximation to the posterior. We will provide a more comprehensive overview of Bayesian deep learning in Section 2.3, with particular focus on the details of SWA-Gaussian in Section 2.4.1.

1.2 Objectives and Contributions

In reality, neural network loss surfaces are complex high-dimensional objects to which low-dimensional intuition rarely applies directly - their nature is an exciting and active area of research. While the broad concepts and intuitions presented in Section 1.1 are widely believed to have some basis in reality [42, 35], the particular role which these distinct notions of uncertainty play in overall uncertainty representation is not well understood. This work focuses on an exploration and comparison of deep ensembles and SWA-Gaussian, as "champions" of global and local uncertainty, respectively. Ultimately we aim to clearly present empirical findings on their differences, similarities, and trade-offs. We achieve this by conducting a number of experiments, focusing on four architecture-dataset pairings in the setting of multiclass image classification.

While we do not conclusively delineate the roles of local and global uncertainty within this work, we do conduct an extensive empirical investigation using SWA-Gaussian and deep ensembles:

- We reimplement deep ensembles [38], SWA [32], MultiSWA [61], unimodal SWA-Gaussian [43], and MultiSWAG [61] for probabilistic classification.
- We compare the storage and performance trade-offs of deep ensembles versus unimodal SWA-Gaussian on a number of model architectures and datasets, providing practical recommendations based on our analyses.
- We empirically demonstrate that representing local uncertainty leads to better calibration, whereas representing global solutions leads to predictive performance increases.
- We investigate the role of diversity as it relates to local and global uncertainty, finding that models from different modes tend to be more diverse than models from the same mode. We find that the balance between diversity and predictive performance is nuanced in general.
- We empirically find that ensembling global solutions does not necessarily improve in-distribution model calibration, leading to significant underconfidence.

Chapter 2

Background and Related Work

In this chapter, we begin by reviewing the particular importance of representing both aleatoric and epistemic uncertainty in deep learning. We then introduce the setting of probabilistic supervised learning as a general framework for learning a representation of predictive uncertainty. Following this, we provide an overview of probabilistic modelling from the Bayesian perspective, prior to giving a succinct review of methods for approximate inference and motivating our interest in SWA-Gaussian. Finally, we revise deep ensembles and their compatibility with the Bayesian perspective, before addressing MultiSWAG as the prototypical method for modelling both local and global epistemic uncertainty.

2.1 Modelling Uncertainty

Given a set of possible observations $\mathcal{X}$, and a set of possible labels $\mathcal{Y}$ to assign, often we would like to learn a relationship between the the two sets. Given some dataset of observations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m \in (\mathcal{X} \times \mathcal{Y})^m$, we might try to formulate a classical supervised learning problem, using the data and our own inductive biases to algorithmically learn a predictive function $f : \mathcal{X} \to \mathcal{Y}$ which assigns labels to the examples $x \in \mathcal{X}$ with low "error" as defined by some loss function. However, often the underlying process mapping from $\mathcal{X}$ to $\mathcal{Y}$ will not perfectly correspond to a functional relationship. We may for example observe $(x_1, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$ with $x_1 = x_2$ and $y_1 \neq y_2$. This may be due to a misrepresentation of the space of examples $\mathcal{X}$, i.e. through measurement noise or missing information, or it may simply be an artefact of a truly random underlying process. Regardless, we cannot realistically expect any learning algorithm to produce a perfect deterministic model of real-world data. Probability theory allows us to relax our assumptions: rather than learning to pick a single label $f(x) \in \mathcal{Y}$ for any given observation $x \in \mathcal{X}$, we could instead learn to produce a distribution over the space of labels $p(x) \in Distr(\mathcal{Y})$. This allows our model to fully represent its expectations about the space $\mathcal{Y}|\mathcal{X} = x$. We avoid forcing our model to collapse what it has learned into a single point mass in $\mathcal{Y}$, ultimately retaining more information which we can use to make intelligent decisions.

2.1.1 Why Model Predictive Uncertainty in Deep Learning?

Traditionally, deep neural networks have not been used to explicitly attempt to model uncertainty [34]. They directly modelled functions $f : \mathcal{X} \to \mathcal{Y}$ instead of probability distributions $p : \mathcal{X} \to Distr(\mathcal{Y})$. This is especially apparent in regression settings for example, where a neural network typically only has a single output node for a real-valued response. Neural networks for classification, on the other hand, often do give normalised score vectors as a means to arrive at a final prediction, though these scores are usually only encouraged to be maximal at the best prediction, and not necessarily to capture a useful notion of uncertainty.

The fundamental problem with this approach comes about when the predictions of neural networks are used as part of a wider decision-making system. Neural networks are notoriously powerful models, able to achieve human-level predictive accuracy on some tasks. These models are being applied in the domains of medical diagonistics [16, 63], automated vehicles [8], manufacturing [58], to name but a few.

However, as we have discussed, we cannot expect any model of real-world data to be perfect in every instance. We must ultimately account for failure. This is not just a theoretical concern: recently, there have been several controversial examples of failure related to erroneous predictions in artificially intelligent systems in industry:

- In May 2016 a Tesla Model S collided with a tractor trailer in Florida, resulting in fatal injuries to the Tesla driver. Both the automated driving system and the driver failed to take action to avoid the collision, despite visibility of the trailer for at least 7 seconds prior [1]. The system failed due to misidentifying the white side of the trailer for bright sky [34].
- Another tragic example relating to automated vehicles occurred in March 2018, when the first pedestrian fatality occurred due to a self-driving vehicle. According to data obtained from Uber's self-driving system, about 6 seconds before impact, the perception system cycled through classifications of the pedestrian as an unknown object, then a vehicle, and then a bicycle, with varying predicted trajectories [2].
- In medicine, IBM's Watson for Oncology was criticised by medical practitioners for offering what were, in their expert view, multiple examples of unsafe and incorrect treatment recommendations [57].

Explicit predictive uncertainty representation is therefore extremely important in deep learning (and indeed any form of predictive modelling). If systems or agents such as those in the above examples are imbued with good representations of their own uncertainty, then undesirable outcomes can be more actively avoided by acting according to uncertainty, i.e. perhaps by motivating more cautious action, or relaying to human experts.

2.1.2 Aleatoric and Epistemic Uncertainty

Broadly speaking, we may group sources of predictive uncertainty in deep learning into *aleatoric* uncertainties, and *epistemic* uncertainties [34]. Let us suppose we have a θ-parameterised neural network model $p_\theta : \mathcal{X} \to Distr(\mathcal{Y})$ for the predictive distribution $\mathbb{P}(\mathcal{Y}|\mathcal{X})$. In this context, the *aleatoric* uncertainty is the uncertainty inherent to this true predictive distribution. If $\mathbb{P}(\mathcal{Y} = y_1|\mathcal{X} = x) = 0.99999$, then qualitatively we have low aleatoric uncertainty as we can be almost certain that the assignment for x will be y_1. If our model is perfect, i.e. if $p_\theta \equiv \mathbb{P}(\mathcal{Y}|\mathcal{X})$, then our predictive uncertainty *is* the aleatoric uncertainty. In practice though, attempting to build a model introduces *epistemic* uncertainty. In this context, epistemic uncertainty is uncertainty about the model of the distribution. If we cannot be certain that $p_\theta \equiv \mathbb{P}(\mathcal{Y}|\mathcal{X})$, then we have epistemic uncertainty of one form or another. At the highest level, we may have epistemic uncertainty in the use of a neural network over a linear model or support vector machine or any other type of model. We may then have uncertainty in the form of our neural network's architecture. In practice, we are usually interested in the parametric uncertainty of a neural network, i.e. the epistemic uncertainty in the parameter values of the weights of the network's connections.

Local and Global Uncertainty are Epistemic

Local and global uncertainty, as discussed in the introduction, each correspond to a particular characterisation of parametric uncertainty. Analytically, global uncertainty in our neural network's parameters corresponds to the idea that many settings $p_{\theta 1}$, $p_{\theta 2}$, $p_{\theta 3}$, etc. may correspond to good models for the observed data. Local uncertainty captures that for any one of these sets of parameters, say θ_1, we can also typically choose $\theta_1 + \epsilon$ for some small $\epsilon \in \mathbb{R}^{|\theta|}$ and obtain a more or less equally good model for the data. SWA-Gaussian, deep ensembles, and the variants investigated in this work are algorithms for building models which account for these types of parametric uncertainty.

Handling Aleatoric and Epistemic Uncertainty

We "handle" aleatoric uncertainty by deciding to try to model it. That is, rather than constructing a function $f : \mathcal{X} \to \mathcal{Y}$, we try to learn a predictive functional $p : \mathcal{X} \to Distr(\mathcal{Y})$, as we have discussed. In Section 2.2, we will provide a brief overview of the *probabilistic supervised learning* paradigm, focusing on discussing how to construct neural network models which represent aleatoric uncertainty, without focusing on the epistemic uncertainty this induces.

To handle the epistemic component of a model's predictive uncertainty, we require significant modifications to the classical training procedure. In Section 2.3, we will cover probabilistic (Bayesian) modelling as a framework for handling epistemic uncertainty in our models. This will put us in a position to review the methods that we ultimately investigate in this work.

2.2 Probabilistic Supervised Learning

In order to discuss modelling epistemic uncertainty, which is the focus of this work, we first need to touch upon *attempting* to model aleatoric uncertainty. Probabilistic supervised learning [23] is the gemeral framework for doing this, wherein we aim to learn the predictive distribution $\mathbb{P}(\mathcal{Y}|\mathcal{X})$ as a function of x. Given an input $x \in \mathcal{X}$, this enables a predictive model to assign confidence to the output space $\mathcal{Y}$ through the language of probability. We will introduce the setting of probabilistic supervised learning pragmatically through the empirical risk minimisation algorithm, before discussing the practicalities of modelling distributions with neural networks, and ultimately *why* we need to go further to model epistemic uncertainty.

2.2.1 Empirical Risk Minimisation in the Probabilistic Setting

Suppose we are given a dataset of input-output pairs $\mathcal{D} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^m$, assumed to be samples from some "true" distribution $\mathbb{P}(\mathcal{X} \times \mathcal{Y})$. Recall the empirical risk minimisation algorithm from classical supervised learning: we select some hypothesis class $\mathcal{H} \subset \mathcal{X} \to \mathcal{Y}$, and aim to choose $f \in \mathcal{H}$ with minimal *empirical risk*:

$$\mathcal{E}_{\mathcal{D}}(f) = \frac{1}{m} \sum_{i=1}^{m} L(f(x_i), y_i)$$

Here, $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is some loss function describing a suitable notion of error in the output space. The idea here is that so long as the space $\mathcal{H}$ is constrained enough to avoid overfitting (possibly through additional regularisation), the empirical risk acts as a proxy for the *true risk*, which we ultimately desire to minimise:

$$\mathcal{E}(f) = \mathbb{E}[L(f(x), y)] = \int_{(x,y) \in \mathcal{X} \times \mathcal{Y}} L(f(x), y) d\mathbb{P}(\mathcal{X} \times \mathcal{Y})$$

Empirical risk minimisation in the probabilistic supervised setting takes much the same form, except rather than a space of predictive *functions* $\mathcal{H} \subset \mathcal{X} \to \mathcal{Y}$, we have a space of predictive *functionals* $\mathcal{H} \subset \mathcal{X} \to Distr(\mathcal{Y})$. This in turn means that in the above definitions we must replace the loss function L with a loss *functional* $\mathcal{L} : Distr(\mathcal{Y}) \times \mathcal{Y} \to \mathbb{R}^+$, describing a notion of error between a distribution and an observed sample, i.e. for $p \in \mathcal{H}$:

$$\mathcal{E}(p) = \mathbb{E}[\mathcal{L}(p(x), y)] \qquad \mathcal{E}_{\mathcal{D}}(p) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f(x_i), y_i)$$

In the classical setting it is typical to select a loss function to encourage learning different notions of minimiser, e.g. when the loss is $L(y, \hat{f}(x)) = (y - \hat{f}(x))^2$

the true risk is minimised when $\hat{f}(x)$ is the mean of the predictive distribution, whereas for the absolute loss $L(y, \hat{f}(x)) = |y - \hat{f}(x)|$ the true risk is minimised by the median. By contrast, in the probabilistic setting, the choice of loss functional $\mathcal{L}$ is somewhat reduced as a free parameter since we are invariably interested in finding the predictive distribution $\mathbb{P}(\mathcal{Y}|\mathcal{X} = x)$ as a function of $x \in \mathcal{X}$. Thus, we ought to have that the setting $p(x) = \mathbb{P}(\mathcal{Y}|\mathcal{X} = x)$ minimises the true risk. The class of loss functionals for which this is the case are called *proper* losses. The class of loss functionals for which $\mathbb{P}(\mathcal{Y}|\mathcal{X} = x)$ is a unique minimiser are called *strictly proper* losses [22, 23].

The Negative Logarithmic Loss Functional

One strictly proper loss functional of particular importance is the *negative logarithmic loss*, defined simply as:

$$\mathcal{L}(p(x), y) = -\log p(x)(y)$$

Where the notation $p(x)(y)$ corresponds to the probability mass/density function $p(x) \in (\mathcal{Y} \to \mathbb{R}^+)$ evaluated at $y \in \mathcal{Y}$. Interestingly, the logarithmic loss is the only loss functional which is in general both strictly proper and *strictly local* [6], meaning it can be directly expressed as a function of $p(x)(y)$ only. This is a subtly important property, as it means that it is the only strictly proper loss functional that can be evaluated for a given sample $(x, y) \in \mathcal{X} \times \mathcal{Y}$ without knowing anything else about the distribution $p(x)$ other than the mass/density it assigns to y. This makes the logarithmic loss computationally efficient (assuming that the mass/density function is), and also theoretically desirable.

Unsurprisingly, due to the monotonicity of the logarithm, empirical risk minimisation with the negative logarithmic loss is equivalent to maximum likelihood estimation over the hypotheses in $\mathcal{H}$:

$$\begin{aligned}\operatorname*{argmin}_{p \in \mathcal{H}} \left[\frac{1}{m} \sum_{i=1}^{m} -\log(p(x_i)(y_i)) \right] &= \operatorname*{argmin}_{p \in \mathcal{H}} \left[-\log \left(\prod_{i=1}^{m} p(x_i)(y_i) \right) \right] \\ &= \operatorname*{argmax}_{p \in \mathcal{H}} \prod_{i=1}^{m} p(x_i)(y_i)\end{aligned}$$

As such, the negative logarithmic loss is essentially canonical for probabilistic supervised learning, and is particularly important for Bayesian deep learning, as we shall see in Section 2.3.

2.2.2 Learning Distributions with Neural Networks

We began this chapter by motivating the modelling of probability distributions with neural networks. In practice, this is achieved by fixing a parametric form for the density/mass function, and representing any distributional parameters as the outputs of a suitable neural network architecture. We will explicitly cover (Gaussian) regerssion and classification, as the latter will be the focus of our experiments in Section 3.

Regression

In the regression setting, usually we assume that the predictive distribution is Gaussian, i.e. $\mathbb{P}(\mathcal{Y}|\mathcal{X} = x) \equiv \mathcal{N}(\mu(x), \sigma^2(x))$. Thus, we construct a neural network architecture parameterised by θ with two output nodes, one corresponding to $\mu_\theta(x)$ and the other to $\sigma^2_\theta(x)$. Note that we may also choose to make σ independent of x (homoscedastic), in which case it would be parameterised as a learned constant in θ. The Gaussian density functional represented by such a network $p_\theta : \mathcal{X} \to (\mathcal{Y} \to \mathbb{R}^+)$ then takes the form:

$$p_\theta(x)(y) = \frac{1}{\sigma_\theta(x)\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y - \mu_\theta(x)}{\sigma_\theta(x)}\right)^2\right)$$

And so the negative logarithmic loss corresponds to:

$$\begin{aligned} -\log(p_\theta(x)(y)) &= -\log\left(\frac{1}{\sigma_\theta(x)\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y - \mu_\theta(x)}{\sigma_\theta(x)}\right)^2\right)\right) \\ &= \log(\sigma_\theta(x)\sqrt{2\pi}) + \frac{1}{2}\left(\frac{y - \mu_\theta(x)}{\sigma_\theta(x)}\right)^2 \\ &= \frac{(y - \mu_\theta(x))^2}{2\sigma^2_\theta(x)} + \frac{\log \sigma^2_\theta(x)}{2} + const. \end{aligned}$$

Hence, given a dataset $\mathcal{D} = \{(x_i, y_i\}_{i=1}^m$ our minimisation objective for ERM is:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{m} \left[\frac{(y_i - \mu_\theta(x_i))^2}{\sigma^2_\theta(x_i)} + \log \sigma^2_\theta(x_i)\right]$$

As an aside, notice that if we assume homoscedasticity with respect to x, this becomes the standard MSE objective for regression problems.

If we were to instead assume the predictive distribution to be Laplacian, we would obtain the MAE objective. Thus, the nature of the choices of loss function in the classical setting correspond to different assumptions about the shape of the underlying distribution in the more general probabilistic setting (see Proposition 2.9 in [23]).

Classification

In the classification setting, $\mathcal{Y}$ corresponds to a finite set of classes $\mathcal{Y} = \{c_1, \ldots, c_n\}$ with $\sum_{i=1}^{n} \mathbb{P}(\mathcal{Y} = c_i | \mathcal{X} = x) = 1$ for all $x \in \mathcal{X}$. As such, we typically construct a neural network architecture with n output neurons, one for each class. This allows us to learn the predictive distribution by learning to represent a mass function for any given $x \in \mathcal{X}$, i.e. the i^{th} output node evaluates to $p_\theta(x)(c_i)$. In this setting, given a dataset $\mathcal{D} = \{(x_k, y_k\}_{k=1}^{m}$, the negative logarithmic loss does not simplify beyond its canonical form:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \frac{1}{m} \sum_{k=1}^{m} \log(p_\theta(x_k)(y_k))$$

Note that to obtain a valid distribution, it is conventional to use the softmax activation function to enforce the functions representable by the neural network to satisfy the constraint $\sum_{i=1}^{n} p_\theta(x)(c_i) = 1$ for all $x \in \mathcal{X}$. For a vector $y \in \mathbb{R}^n$:

$$\operatorname{softmax}(y) = \begin{pmatrix} \frac{e^{y_1}}{Z} \\ \vdots \\ \frac{e^{y_n}}{Z} \end{pmatrix} \text{ with } Z = \sum_{i=1}^{n} e^{y_i}$$

Not only does this force the output vector to be normalized, it also allows the network to work in logarithms, which is generally numerically advantageous, in particular helping to avoid vanishing gradients during stochastic gradient descent.

In theory, such a network could then be trained as usual to estimate these distributional parameters as a function of $x \in \mathcal{X}$ via (approximate, possibly regularised) empirical risk minimisation in the form of stochastic gradient descent. However, employing only this classical training procedure leads to problems in modern neural networks, as we shall now see.

2.2.3 Issues with Overconfidence in ERM

We have thus introduced how to model simple probability distributions using neural networks, as well as the empirical risk minimisation algorithm using the logarithmic loss, with all of its satisfying theoretical properties. Could we not then simply train neural networks to learn a good approximation to the true predictive distribution by minimising their empirical risk through stochastic gradient descent, as we do in the classical setting?

In fact, in the classification setting, the general form of the architecture we use to model categorical probability distributions is exactly the same as those we classically use to perform predictive modelling. What's more, we also typically use the negative logarithmic loss function to train, since theoretically it should also lead to the best predictions. The only difference is that in a purely predictive neural network model, we simply take the argmax of the probability vector to compute our prediction and typically discard the rest of the information. This allowed Guo et al. (2017) [24] to go back and train classification architectures that had previously been used as predictive models in order to explicitly evaluate them as probabilistic models - and to their surprise they found that while older neural network architectures were producing reasonable uncertainty estimates, modern architectures (within the last 10, now 13 years) had steadily become more and more *overconfident*. Guo et al. [24] identify a number of modern practices that have lead to the overconfidence of modern neural architectures. One of the most surprising is the negative logarithmic loss itself: It is possible to overfit to the NLL without overfitting to the 0-1 loss. Empirically, they demonstrate that networks benefit from overfitting to NLL with an *increase in test accuracy*, and thus modern neural networks with extremely high capacity are able to increase their predictive performance essentially by training to be overconfident. This effect was observed both on examples coming from the modelled distribution (in-distribution), and from outside of it (out-distribution) - meaning if a network was trained on pictures of and then showed a picture of a dog, it would still classify the dog as a type of tree with high confidence! This overconfidence phenomenon of course is dangerous in practice if we desire to use a neural net to make predictions that could have serious consequences, as in the examples we showed earlier.

Measuring Overconfidence

Guo et al. (2017) primarily used *reliability* diagrams [48], as well as *expected calibration error* (ECE) as meaningful metrics with which to measure the quality of a model's uncertainty estimates. We illustrate an example reliability diagram in Figure 2.1

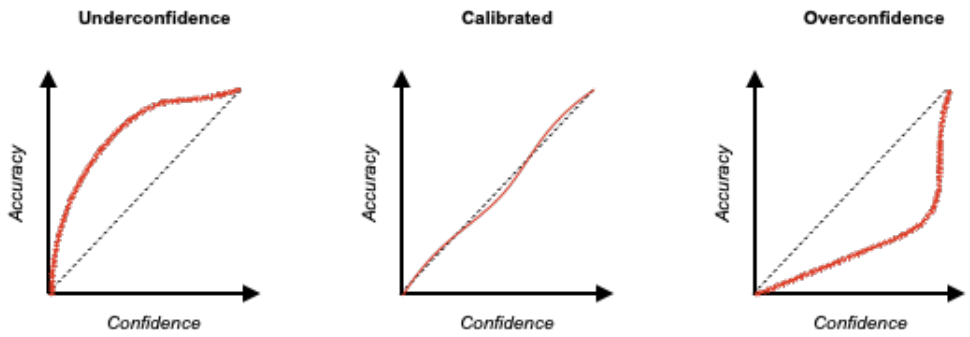


Figure 2.1: Illustrations of reliability diagrams corresponding to an underconfident model, a *calibrated* model, and an overconfident model. The red line depicts each classification model's observed accuracy against its confidence (estimated class probability).

Of course, in practice it is not possible to directly plot accuracy as a function of model confidence. Instead, approximate reliability diagrams are plotted over a held out dataset by binning examples according to the maximum class probability. This maximum class probability is the model's confidence in its optimal prediction. A model's accuracy can then be calculated within each bin, so that we may approximate the reliability curve. In a perfectly *calibrated* model, the model confidence should be equal to its accuracy. Often it is useful to have a scalar measure of calibration: expected calibration error [45] is simplest to interpret as the area between the line $y = x$ and the reliability curve. Thus, ECE is a useful scalar measure which can also be approximated using the aforementioned binning technique. We will make extensive use of both of these in Section 3.

Reasons for Overconfidence

The fundamental reason for overconfidence, regardless of the modern techniques that we apply that implicitly encourage it, is a lack of epistemic uncertainty representation [61, 19, 34]. As we discussed in the introduction, neural network loss surfaces are multimodal, meaning that many parameter settings are consistent with the training data. Selecting a single parameter setting and ignoring all others through classical empirical risk minimisation, we ignore the vast amount of information we have about other possible solutions encoded in the loss surface. For the remainder of this work, we will therefore focus on representing epistemic uncertainty in deep learning via Bayesian learning.

2.3 Bayesian Deep Learning

Bayesian modelling is a powerful technique for handling epistemic uncertainty, providing machinery for us to express and reason about uncertainty within a space of possible models. In this section, we will first provide an overview of Bayesian inference, before discussing its application in the context of deep learning.

Bayesian Inference

Bayesian probability theory quantifies a "degree of belief" consistent with the laws of probability theory. We can use this interpretation of probability to quantify our own uncertainty in mathematical statements, despite the fact that often they are either true, or they are not. Bayesian probability is only required to be consistent with respect to the agent whose beliefs it codifies. Bayesian inference is the process of updating these beliefs when faced with new data. In the context of modelling, given a hypothesis class $\mathcal{H}$ containing a set of exhaustive and mutually exclusive models for an underlying data generating process $\mathcal{P}$, the first step in Bayesian inference is to assign a prior probability to the hypothesis class $\mathbb{P}(\mathcal{H})$ representing a belief that the true process is contained within the class, i.e. $\mathcal{P} \in \mathcal{H}$. We may then assign prior belief $\mathbb{P}(p|\mathcal{H})$ to the elements of $\mathcal{H}$ corresponding to our degree of belief that $p \in \mathcal{H}$ is the "true" model $\mathcal{P} = p$. Then, given a set of observed samples from the true distribution $\mathcal{D} \sim \mathcal{P}^m$, we can infer the *posterior* probability $\mathbb{P}(p|\mathcal{D}, \mathcal{H})$ of any given model $p \in \mathcal{H}$ by applying the definition of conditional probability and the law of total probability to formulate *Bayes' rule*:

$$\mathbb{P}(p|\mathcal{D}, \mathcal{H}) = \frac{\mathbb{P}(p, \mathcal{D}|\mathcal{H})}{\mathbb{P}(\mathcal{D}|\mathcal{H})} = \frac{\mathbb{P}(\mathcal{D}|p, \mathcal{H})\mathbb{P}(p|\mathcal{H})}{\int_{p\in\mathcal{H}} \mathbb{P}(\mathcal{D}|p, \mathcal{H}) d\mathbb{P}(p|\mathcal{H})}$$

Where we refer to $\mathbb{P}(\mathcal{D}|\mathcal{H}) \equiv \int_{p\in\mathcal{H}} \mathbb{P}(\mathcal{D}|p, \mathcal{H}) d\mathbb{P}(p|\mathcal{H})$ as the *marginal likelihood* or *model evidence*, and to $\mathbb{P}(\mathcal{D}|p, \mathcal{H})$ as the *likelihood*. Note that the probability $\mathbb{P}(\mathcal{H})$ does not affect our calculations within $\mathcal{H}$, and thus we often omit the dependence on $\mathcal{H}$ (or equivalently, presume to wholeheartedly believe our inductive biases and assign $\mathbb{P}(\mathcal{H}) = 1$). We may then write the above more succinctly as:

$$\mathbb{P}(p|\mathcal{D}) = \frac{\mathbb{P}(p, \mathcal{D})}{\mathbb{P}(\mathcal{D})} = \frac{\mathbb{P}(\mathcal{D}|p)\mathbb{P}(p)}{\int_{p\in\mathcal{H}} \mathbb{P}(\mathcal{D}|p) d\mathbb{P}(p)}$$

Bayesian inference is simply the application of the above, used to update prior belief to posterior belief when presented with data. Intuitively, we evaluate the probability of the observed data in any one given reality $p \in \mathcal{H}$ in the context of our prior beliefs, and normalize this probability over all possible realities. Note that in order to achieve this we require a likelihood function to be defined for each $p \in \mathcal{H}$, telling us the probability of observing the data $\mathcal{D}$ given p is the "true" model in $\mathcal{H}$.

2.3.1 Bayesian Model Averaging

Suppose then that we wish to use Bayesian inference to learn a predictive distribution via some neural network architecture $\mathcal{A}$. For simplicity, consider $\mathcal{A}$ as a mapping from some space of parameters $\Theta \subset \mathbb{R}^{|\mathcal{A}|}$ to the space of predictive masses/densities $\mathcal{X} \to (\mathcal{Y} \to \mathbb{R}^+)$. Our hypothesis space therefore takes the form $\mathcal{H}_\mathcal{A} = \{p_\theta \equiv \mathcal{A}(\theta) | \theta \in \Theta\}$. Assuming that $\mathcal{H}_\mathcal{A}$ is *identifiable*[1] with respect to Θ (i.e. $\mathcal{A}$ is injective), we may equivalently consider our hypothesis space as the space of parameters Θ. In this case, given some dataset $\mathcal{D} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^m$ and a prior over Θ, we can obtain the posterior over parameters:

$$\begin{aligned}\mathbb{P}(\theta|\mathcal{D}) &= \frac{\mathbb{P}(\theta, \mathcal{D})}{\mathbb{P}(\mathcal{D})} = \frac{\mathbb{P}(\theta, x_{1:m}, y_{1:m})}{\mathbb{P}(x_{1:m}, y_{1:m})} = \frac{\mathbb{P}(\theta, y_{1:m}|x_{1:m})\cancel{\mathbb{P}(x_{1:m})}}{\mathbb{P}(y_{1:m}|x_{1:m})\cancel{\mathbb{P}(x_{1:m})}} \\ &= \frac{\mathbb{P}(y_{1:m}|\theta, x_{1:m})\mathbb{P}(\theta)}{\int_{\theta\in\Theta}\mathbb{P}(y_{1:m}|\theta, x_{1:m})d\mathbb{P}(\theta)} = \frac{\prod_{i=1}^m p_\theta(y_i|x_i)\mathbb{P}(\theta)}{\int_{\theta\in\Theta}\prod_{i=1}^m p_\theta(y_i|x_i)d\mathbb{P}(\theta)}\end{aligned}$$

Where we have explicitly cancelled the probability of the $\mathcal{X}$ samples to allow us to use our neural network (a *discriminative* model) to compute a likelihood function. Notice that if we take the negative logarithm of the posterior we obtain a quantity that is equivalent to the regularized negative logarithmic loss, up to a constant:

$$-\log\mathbb{P}(\theta|\mathcal{D}) = \sum_{i=1}^m -\log p_\theta(y_i|x_i) - \log\mathbb{P}(\theta) + \log\mathbb{P}(\mathcal{D})$$

where the negative log prior is $-\log\mathbb{P}(\theta) > 0$ since $\mathbb{P}(\theta) \in [0, 1]$, and hence *regularisation* and *prior belief* are equivalent in Bayesian deep learning. Furthermore, we can perform maximum a posteriori estimation via gradient descent over the negative log posterior. However, we have already discussed that this leads to overconfidence. Let's instead suppose then that we can compute this posterior in its entirety. A better idea would then be to marginalise over Θ to obtain a predictive distribution conditioned on the data we have observed so far, thus accounting for epistemic uncertainty:

$$\begin{aligned}\mathbb{P}(y \,|\, x, \mathcal{D}) &= \int_{\theta\in\Theta} \mathbb{P}(y \,|\, x, \theta) d\mathbb{P}(\theta \,|\, \mathcal{D}) \\ &= \int_{\theta\in\Theta} p_\theta(y \,|\, x) d\mathbb{P}(\theta \,|\, \mathcal{D})\end{aligned}$$

[1] Identifiability ensures that each $\theta \in \Theta$ parameter setting refers to a unique predictive distribution $p_\theta \in \mathcal{H}_\mathcal{A}$. It is possible to do Bayesian inference in parameter space without strict identifiability, however the marriage to the space of predictive distributions is more technical.

The resulting predictive distribution depends only on the observed data, combined with our implicit belief that $\mathcal{H}_\mathcal{A}$ is able to represent the "true" predictive distribution. This marginalisation over the posterior is referred to as *Bayesian model averaging*, because by definition we are taking the expectation of the parametric predictive distribution with respect to the posterior. In practice, the analytical form of the posterior will not be available, and the space Θ is often too large for this integral to be computed numerically, so we instead resort to a simple Monte Carlo approximation:

$$\mathbb{P}(y|x, \mathcal{D}) \approx \frac{1}{n} \sum_{i=1}^{n} p_{\theta i}(y|x) \text{ where } \theta_i \sim \mathbb{P}(\theta|\mathcal{D})$$

Intractability

Due to its natural incorporation of parametric uncertainty, Bayesian model averaging is a theoretically desirable algorithm for learning a representation of the true predictive distribution from data (see Figure 2.2 for a visualisation). Unfortunately, performing a true Bayesian model average relies on our ability to obtain and evaluate the posterior distribution $\mathbb{P}(\theta|\mathcal{D})$. For modern neural networks with many thousands or millions of parameters, evaluating the posterior is often not possible. In particular the integral corresponding to the evidence is often computationally intractable. This gives rise to methods for *approximate Bayesian inference* in deep learning, which usually aim to come up with a tractable approximation to the posterior such that we may compute a Bayesian model average.

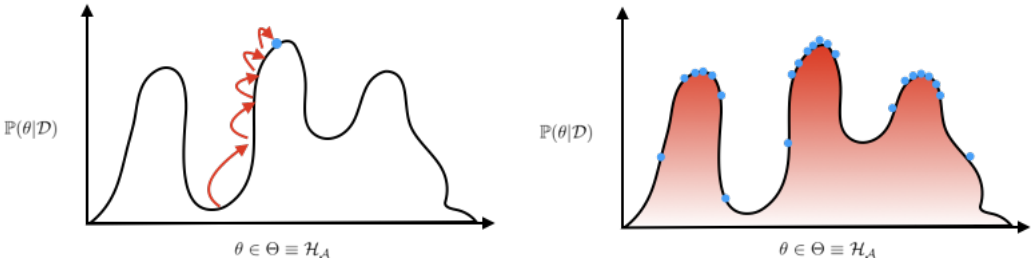


Figure 2.2: A cartoon depicting the conceptual difference between optimization (left) and marginalisation (right).

There are a number of approaches to approximating the posterior distribution over a neural network's parameters so that we may obtain an approximate predictive distribution through Bayesian model averaging. The majority of modern methods fall into two of the wider camps of approximate Bayesian inference: *Variational inference* (VI) and *Markov Chain Monte Carlo* (MCMC). Both happen to be relevant for our discussions on SWA-Gaussian, so we will give a concise overview of each in the context of modern deep learning.

2.3.2 Variational Inference

In variational inference we typically parameterise a class of distributions $q_\psi(\theta)$ over $\theta \in \Theta$ with which we attempt to approximate $\mathbb{P}(\theta|\mathcal{D})$, usually by minimising the (reverse) KL-divergence between the two distributions with respect to ψ.

$$KL(q_\psi(\theta)|\mathbb{P}(\theta|\mathcal{D})) = \mathbb{E}_{q_\psi}\left[\log \frac{q_\psi(\theta)}{\mathbb{P}(\theta|\mathcal{D})}\right] = \int_{\theta\in\Theta} \log \frac{q_\psi(\theta)}{\mathbb{P}(\theta|\mathcal{D})} dq_\psi(\theta)$$

Note that $\mathbb{P}(\theta|\mathcal{D}) = \frac{\mathbb{P}(\theta,\mathcal{D})}{\mathbb{P}(\mathcal{D})}$, and so:

$$\begin{aligned} KL(q_\psi(\theta)|\mathbb{P}(\theta|\mathcal{D})) &= \int_{\theta\in\Theta} \log \frac{q_\psi(\theta)}{\mathbb{P}(\theta,\mathcal{D})} dq_\psi(\theta) + \int_{\theta\in\Theta} \log \mathbb{P}(\mathcal{D}) dq_\psi(\theta) \\ &= \underbrace{\int_{\theta\in\Theta} \log q_\psi(\theta) - \log \mathbb{P}(\theta,\mathcal{D}) dq_\psi(\theta)}_{\mathcal{F}(q_\psi)} + \log \mathbb{P}(\mathcal{D}) \\ \implies \qquad \log \mathbb{P}(\mathcal{D}) &= KL(q_\psi(\theta)|\mathbb{P}(\theta|\mathcal{D})) - \mathcal{F}(q_\psi) \end{aligned}$$

Hence minimising the KL-divergence is equivalent to maximising the *evidence lower bound* $\mathcal{F}(q_\psi) = \mathbb{E}_{q_\psi}[\log q_\psi(\theta) - \log \mathbb{P}(\theta,\mathcal{D})]$. The key to this approach is to choose a class q_ψ which makes optimization of the evidence lower bound objective tractable. The prototypical work on variational inference in deep learning was due to Hinton et al. (1993) [28], wherein they select a Gaussian distribution $q_{\mu,\sigma}(\theta) = \prod_{i=1}^{|\mathcal{A}|} \mathcal{N}(\theta_i|\mu_i,\sigma_i^2)$, i.e. with all network weights presumed independent of one another. For small architectures (i.e. with a single hidden layer), the KL-objective is analytically tractable and thus they were able to train Bayesian neural networks, though with limited practical success due to the assumption of independence between network weights being unrealistic. Through a particular choice of activation function and careful analytic treatment, Barber et al. (1998) [5] extended this method to allow for full covariance matrices to be used, also implementing a version of their algorithm whose rank could be tuned in accordance with storage constraints. Unfortunately, modern day neural networks do not tend to admit tractable integrals for the evidence lower bound. However, variational inference has recently resurfaced as an area of active research. Notably, Blundell et al. (2015) [7] proposed *Bayes by Backprop*, demonstrating the use of the reparameterisation trick [36] to automatically train variational parameters via backpropagation. This has since been applied successfully to both recurrent neural networks [18] and convolutional neural networks [54]. Still, due to the computational cost of modern variational techniques, it remains common to employ either a Gaussian distribution or a relatively small mixture of Gaussians, and thus in practice it is difficult to capture the multimodal posterior globally using variational methods. This perhaps plays a part in Bayesian deep learning methods being traditionally considered as focused on local uncertainty.

2.3.3 Markov Chain Monte Carlo

In the context of Bayesian deep learning, Markov chain Monte Carlo methods aim to construct a Markov chain p whose stationary distribution is equal to the posterior $\mathbb{P}(\theta|\mathcal{D})$. That is, if we sample a sequence from such a chain:

$$\theta_1, \theta_2, \ldots, \theta_n \quad \text{where } \theta_t \sim p(\theta|\theta_{t-1})$$

Then as $n \to \infty$, we have $\mathbb{P}(\theta_n = \theta) \to \mathbb{P}(\theta|\mathcal{D})$. Thus, after an initial "warm-up" period, we can obtain approximate samples from the posterior by sampling from this chain. Metropilis-Hastings [26] (MH) is the canonical algorithm for constructing such a Markov chain. Essentially, MH accepts or rejects generated proposals for the next state in the sequence in a manner that ensures the stationary distribution is the posterior. The key to an effective MCMC method is to maintain a low proposal rejection rate; otherwise sampling from the posterior will be extremely computationally expensive. MCMC approaches to Bayesian deep learning therefore typically focus on constructing efficient proposal distributions.

Hamiltonian Monte Carlo

Modern MCMC-based approaches stem from the seminal work of Neal (1996) [46], who applied *Hamiltonian Monte Carlo* (HMC)[2] to the weights of a neural network. In HMC proposals are generated for MH by starting at the previous parameter state and randomly sampling momentum variables before allowing the laws of Hamiltonian dynamics to play out over a fixed time period[3]. Amazingly, the final state of this procedure has a low rejection rate [47]. Due to its historic effectiveness, HMC is often revered as an example of the potential of Bayesian deep learning. Unfortunately, the discretisation of Hamiltonian dynamics is performed via a hyperparameterised 'leapfrog' method which can be hard to tune. Further, HMC requires full gradient computations [47, 19] and thus is impractical for use with modern (large) networks/datasets. Recently, HMC has been succeeded by Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [12], which attempts to allow for the use of stochastic gradients in the HMC algorithm. However, this introduces approximation errors, and the method remains difficult to tune in practice. Another successor of HMC is Welling and Teh's (2011) Stochastic Gradient Langevin Dynamics [59] (SGLD). The SGLD algorithm deviates from HMC in that it is explicitly based on stochastic gradient descent. However, rather than performing true stochastic optimisation, we inject additional stochasticity in such a way that the algorithm samples from the true posterior, typically by traversing the region around a local mode. Unfortunately, this negates one of the fundamental advantages of HMC, which is its ability to generate distant proposals in practice, and thus escape from any local region of the posterior. A more complete classification of stochastic gradient MCMC algorithms is given in [62].

[2] Alternatively, *Hybrid Monte Carlo* [15] by the original creators of the method.

[3] Neal [47] alikens the proposal sampling mechanism of HMC to a frictionless ice-hockey puck being repeatedly kicked and sliding around the posterior's landscape.

2.3.4 SGD as Approximate Bayesian Inference

For our purposes, the overarching insight in considering Stochastic Gradient MCMC algorithms is the idea that the stochastic gradient descent algorithm itself constitutes a Markov chain. For any given dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{m}$ and neural network p_θ with parameters θ, we perform a stochastic update to move from θ_{t-1} to θ_t:

$$\theta_t := \theta_{t-1} - \eta \sum_{i=1}^{B} \frac{\delta}{\delta\theta} \left[\log(p_\theta(x_i)(y_i)\right]_{\theta=\theta_{t-1}}$$

Thus our next position in parameter space depends only on the current position and the random draw of the B elements of the minibatch. This interpretation also extends to most variants of stochastic gradient descent. In particular, if we consider the Markov chain state to include momentum, then stochastic gradient descent with momentum is also a Markov chain. In theory then, rather than attempting to modify expensive MCMC algorithms to use stochastic gradients effectively, we could attempt to go in the other direction and modify stochastic optimisation algorithms to generate samples from the true posterior.

Constant Learning Rate SGD

Following this perspective, Mandt. et al. (2017) [44] suggest that, given a fixed learning rate, the iterates of stochastic gradient descent converge to a stationary distribution. In particular, under moderate regularity conditions for the local geometry of the loss surface, they show that this stationary distribution is approximately Gaussian. This motivates the idea that fixed learning rate stochastic gradient descent can be used as an approximate Bayesian inference algorithm. Specifically, as a sort of hybrid between variational inference and Markov chain Monte Carlo, wherein we could try to minimise the Kullback-Leibler divergence between the true posterior and the Gaussian stationary distribution of constant learning rate SGD. To this end, by further utilising their regularity assumptions, Mandt et al. (2017) [44] go on to explicitly derive expressions for hyperparameters which minimise this divergence for a number of SGD variants commonly used to train neural networks. Empirically, they find that this perspective is promising, but that the regularity assumptions are likely too strong for their exact theoretical framework to hold.

Mandt et al. [44] conclude with an analysis of iterate averaging in the context of neural networks, and show that under their regularity assumptions, Polyak iterate averaging [51] can lead to an optimal stochastic gradient MCMC sampler. Conceptually, this paves the way for the Stochastic Weight Averaging algorithm [32], which we shall review as a precursor to Stochastic Weight Averaging-Gaussian at the beginning of the next section.

2.4 Local and Global Methods for Uncertainty

In this section, we review the methods which feature in our experiments in Section 3. First, in Section 2.4.1 we describe SWA-Gaussian as a method for capturing local epistemic uncertainty within a given mode. In Section 2.4.2 we provide an overview of deep ensembles, an arguably simpler and more established method which focuses on representing global epistemic uncertainty. Finally, in Section 2.4.3 we discuss the conceptual combination of both methods in MultiSWAG.

2.4.1 Stochastic Weight Averaging-Gaussian (SWAG)

Stochastic Weight Averaging-Gaussian [43] is a recently proposed method for approximate Bayesian inference in deep learning which fits the iterates of stochastic gradient descent within a local mode to a Gaussian distribution. In order to present a sufficiently detailed picture of SWA-Gaussian, we will begin by discussing the Stochastic Weight Averaging algorithm.

Stochastic Weight Averaging (SWA)

The key motivation presented by Izmailov et al. [32] for the Stochastic Weight Averaging algorithm is that parameters identified by SGD tend to lie on the boundary of a region of low loss solutions at the end of training. They observe that continuing to run SGD with a constant learning rate allows SGD to explore the boundary around a local mode, reminiscent of the stationary distribution discussed in Section 2.3.4. Izmailov et al. demonstrate that averaging these SGD iterates in parameter space brings us to a more central point in this low loss region, resulting in a broader optimum that is more robust to the shift between training data and unseen data. This intuitive idea that broader optima result in better generalisation has been empirically validated in e.g. [35]. We reproduce the SWA algorithm in Algorithm 1, and present an illustration in Figure 2.3.

Algorithm 1: Constant LR SWA

Input: Pretrained Model θ_0, Number of iterations T, Update frequency c, Learning Rate η

Init: $\theta_{\text{SWA}} = \theta_0$, $n = 1$

for $t \in \{1, \dots T\}$ **do**

  $\theta_t = \theta_{t-1} - \eta \nabla \mathcal{L}_\theta(\theta_{t-1})$ {Perform SGD Update} ;

  if t mod $c == 0$ **then**

    $\theta_{\text{SWA}} = \frac{n}{n+1}\theta_{\text{SWA}} + \frac{1}{n+1}\theta_t$ {Update SWA} ;

    $n = n + 1$ {Update Number of Models} ;

end

Return: θ_{SWA}

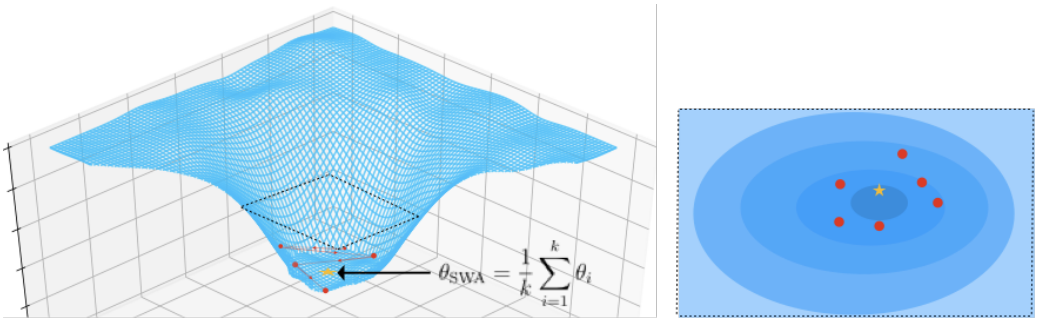


Figure 2.3: An illustration depicting SGD iterates lying on the boundary (red) averaging to give the SWA solution (gold).

From SWA to SWA-Gaussian

SWA-Gaussian [43] is an extension of SWA, motivated by the idea that SGD converges to a stationary distribution [44]. Maddox et al. view SWA as an estimate of the mean of this stationary distribution, and propose to also estimate its covariance structure in order to define an explicit Gaussian approximation to the posterior within a given mode. Since the storage cost of a full rank covariance matrix would be quadratic in the number of model parameters, SWA-Gaussian maintains scalability by using a low-rank plus diagonal structure, making it practical even for large neural networks.

The approximate posterior distribution constructed by the SWA-Gaussian algorithm can be written as $\mathcal{N}_{|\theta|}(\theta_{\text{SWA}}, \frac{1}{2}(\Sigma_{\text{diag}} + \Sigma_{\text{low rank}}))$, where Σ_{diag} is a diagonal matrix, and $\Sigma_{\text{low rank}}$ has rank k, where k is a hyperparameter of the method. The algorithm obtains samples in parameter space exactly as in Stochastic Weight Averaging, by starting from a converged solution and using a constant learning rate to continue to explore in the vicinity of a local mode.

The θ_{SWA} term is calculated as in Algorithm 1. The diagonal matrix is an estimate of the (axis-aligned) variance in parameter space. Thus, using the standard identity $\text{Var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$, SWAG keeps track of the mean of the element-wise squared parameter samples $\overline{\theta^2} = \frac{1}{T}\sum_{t=1}^{T}\theta_i^2$, and computes $\Sigma_{\text{diag}} = \overline{\theta^2} - \theta_{\text{SWA}}^2$ at the end of training.

The rank k matrix $\Sigma_{\text{low rank}}$ is a rank k estimate of the covariance matrix. For T parameter samples, the unbiased sample covariance matrix can be written in terms of the deviations d_t from the mean, i.e.

$$\frac{1}{T-1}\sum_{t=1}^{T} d_t d_t^T = \frac{1}{T-1}\sum_{t=1}^{T}(\theta_t - \theta_{\text{SWA}})(\theta_t - \theta_{\text{SWA}})^T$$

In order to fix the rank of $\Sigma_{\text{low rank}}$ at k, SWAG only keeps track of the previous k deviations. Since we do not have access to the true sample mean θ_{SWA} during

training, SWAG further approximates with the current running mean $\overline{\theta}_t$. Thus, SWAG's $\Sigma_{\text{low rank}}$ can be written as:

$$\Sigma_{\text{low rank}} = \frac{1}{k-1} \sum_{t=T-(k-1)}^{T} \hat{d}_t \hat{d}_t^T = \sum_{t=T-(k-1)}^{T} (\theta_t - \overline{\theta}_t)(\theta_t - \overline{\theta}_t)^T$$

Calculating the low-rank matrix in this way allows the algorithm to be memory efficient during training as it keeps the training cost independent of the number of parameter samples. To compute θ_{SWA}, Σ_{diag}, and $\Sigma_{\text{low rank}}$, we need only keep track of the running average $\overline{\theta}_t$, the running average of squares $\overline{\theta^2}_t$, and the previous k deviations $\{\hat{d}_{t-(k-1)}, \ldots, \hat{d}_t\}$ which are stored as a matrix $\hat{D} \in \mathbb{R}^{|\theta| \times k}$. Thus the maximum runtime storage cost of SWAG is the same as its final memory cost $O(k|\theta|)$. Note also that to sample from SWAG, we do not actually need to bother with computing $\Sigma_{\text{low rank}} = \hat{D}\hat{D}^T$ - we can instead use the identity:

$$X \sim \mathcal{N}_m(\mu, \Sigma) \iff X = \mu + \Sigma^{\frac{1}{2}} Z$$

where $\Sigma = \Sigma^{\frac{1}{2}} \Sigma^{\frac{1}{2}T}$ and $Z \sim \mathcal{N}_m(0, I_m)$. Coupled with the closure of Gaussian distributions under addition, this allows us to arrive at:

$$\theta \sim \mathcal{N}_{|\theta|}(\theta_{\text{SWA}}, \frac{1}{2}(\Sigma_{\text{diag}} + \Sigma_{\text{low rank}})) \iff \theta = \theta_{\text{SWA}} + \Sigma_{\text{diag}}^{\frac{1}{2}} Z_{\text{diag}} + \hat{D} Z_{\text{low rank}}$$

where $Z_{\text{diag}} \sim \mathcal{N}_{|\theta|}(0, I_{|\theta|})$ and $Z_{\text{low rank}} \sim \mathcal{N}_k(0, I_k)$. The expression on the right can be calculated in $O(k\theta)$ time, thus drawing one sample from SWAG costs $O(k|\theta|)$ in computation. For completeness, we reproduce the details of the SWAG training procedure as presented by Maddox et al. [43] in Algorithm 2.

Algorithm 2: Constant LR SWA-Gaussian

Input: Pretrained Model θ_0, Number of iterations T, Update frequency c, Learning Rate η, Rank k

Init: $\bar{\theta} = \theta_0$, $\overline{\theta^2} = \theta_0^2$, $\hat{D} = [\,]$, $n = 1$ **for** $t \in \{1, \ldots T\}$ **do**

- $\theta_t = \theta_{t-1} - \eta \nabla \mathcal{L}_\theta(\theta_{t-1})$ {Perform SGD Update} ;
- **if** t mod $c == 0$ **then**
  - $\bar{\theta} = \frac{n}{n+1}\bar{\theta} + \frac{1}{n+1}\theta_t$ {Update First Moment} ;
  - $\overline{\theta^2} = \frac{n}{n+1}\overline{\theta^2} + \frac{1}{n+1}\theta_t^2$ {Update Second Moment} ;
  - **if** len$(\hat{D}) == k$ **then**
    - $\hat{D}$.pop() {Remove Oldest Deviation} ;
  - $\hat{D}$.append$(\theta_t - \bar{\theta})$ {Append Current Deviation};
  - $n = n + 1$ {Update Number of Models} ;

end

Return: $\theta_{\text{SWA}} = \bar{\theta}$, $\Sigma_{diag} = \bar{\theta}^2 - \overline{\theta^2}$, $\hat{D}$

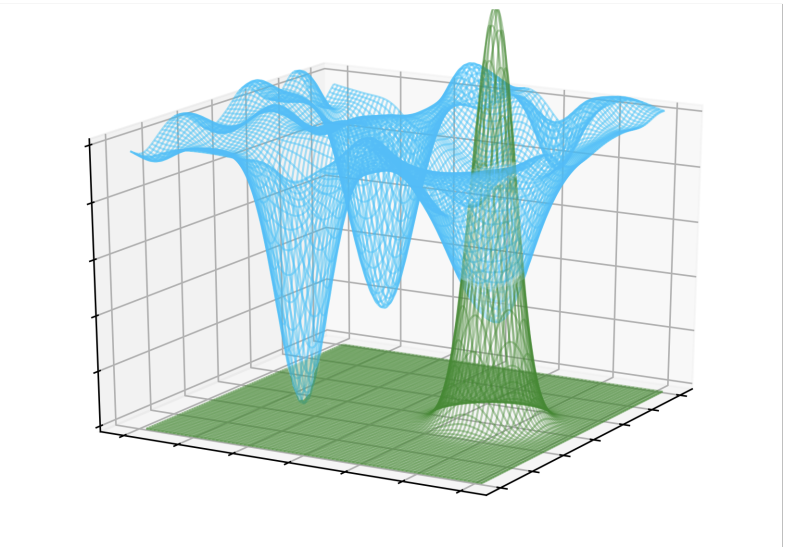


Figure 2.4: An illustration of a SWAG approximation (green) overlayed on the posterior loss surface (blue).

Further Details on SWAG

It is important to note that Maddox et al. [43] demonstrated that the original regularity assumptions motivating the SWAG algorithm, i.e. those made by Mandt et al. [44], do not hold in practice. In particular, they show in accordance with [20, 13] that the posterior loss surface is often relatively flat in some directions surrounding a given optima, and thus ultimately non-quadratic - an assumption that was relied on heavily in the original derivation of the optimal hyperparameters to use for constant learning-rate SGD to obtain the best variational stationary distribution. They did however find that empirically approximating the subspace spanned by SGD iterates with a Gaussian distribution worked in practice, though the hyperparameters of the optimizer had to be tuned manually as in classical training. The use of the covariance approximation $\frac{1}{2}(\Sigma_{\text{diag}} + \Sigma_{\text{low rank}})$ may also seem a bit strange at first sight. After all, the diagonals (variances) should also be approximated in the low rank matrix, and so this decomposition has the effect of halving the non-diagonal elements of the matrix and keeping the diagonals constant. This also makes the SWAG approximation slightly more contrived in its geometrical interpretation with respect to the actual covariance of the SGD iterates. Ultimately SWAG is an empirical method - it is perhaps better to think of it more generally as a Gaussian approximation around the SWA solution with a covariance matrix of $\alpha\Sigma_{\text{diag}} + \beta\Sigma_{\text{low rank}}$, i.e. a variability in the axis-aligned directions, and a variability in the subspace formed by SGD iterates. These α and β are hyperparameters of the method which can be considered as variational parameters which we can use to tune the Gaussian approximation to match the local posterior. It just so happens that $\alpha = \beta = 0.5$ are good settings for the parameters. We will discuss this more in Section 3.

2.4.2 Deep Ensembles

It is essentially common knowledge in the machine learning community that combining multiple models tends to lead to substantial improvements in predictive performance. Indeed, deep neural networks are no exception to this rule, with foundational work on *deep ensembles* dating back to at least the early 90s [25, 21, 50]. Only recently, however, have deep ensembles been proposed as a practical alternative [38] to Bayesian methods for predictive uncertainty representation in deep learning. As we discussed in Sections 2.3.2 and 2.3.3, most traditional Bayesian methods for deep learning require either significant modifications to the training procedure, or additional parameters to be estimated, or both. This can require significant overhead and quickly make training impractical for larger networks. By contrast, if we are able to train a single network, then we are able to train an ensemble of networks, and furthermore we can do so in parallel since each network's parameters do not functionally depend on one another. Thus, conceptual simplicity and scalability are the fundamental advantages of deep ensembles.

Deep Ensemble Algorithm

The predictive distribution of a deep ensemble as presented by Lakshminarayanan et al. [38] corresponds to a uniformly-weighted mixture of the predictive distributions of its members. First, we train a deep ensemble as in Algorithm 3.

Algorithm 3: Deep Ensemble

Input: Random initialisations $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_m$, stochastic optimizer `opt`, number of training iterations T
Init: $\bar{\theta} = [\ \ ]$
for $k \in \{1, \ldots, m\}$ **do**
  $\theta_0 = \hat{\theta}_k$;
  for $t \in \{1, \ldots T\}$ **do**
    $\theta_t = \texttt{opt}(\theta_{t-1}, \nabla \mathcal{L}_\theta(\theta_{t-1}))$ {Train Model} ;
  end
  $\bar{\theta}\,.\text{append}(\theta_T)$ {Add Model to Ensemble};
end
Return: $\bar{\theta}$

Then, given our collection of trained parameters $\bar{\theta} = [\theta_1, \theta_2, \ldots, \theta_m]$, we compute the predictive mass/density at some datapoint (x, y) as:

$$p_{\bar{\theta}}(y|x) = \frac{1}{m} \sum_{i=1}^{m} p_{\theta_i}(y|x)$$

Note that for the categorical distributions we look at in Section 3, this corresponds to simply averaging the predicted probability over each of the ensemble members for each category.

The combination of a random initialisation in parameter space and a stochastic optimization trajectory encourages ensemble members to occupy different modes on the loss surface, as illustrated in Figure 2.5.

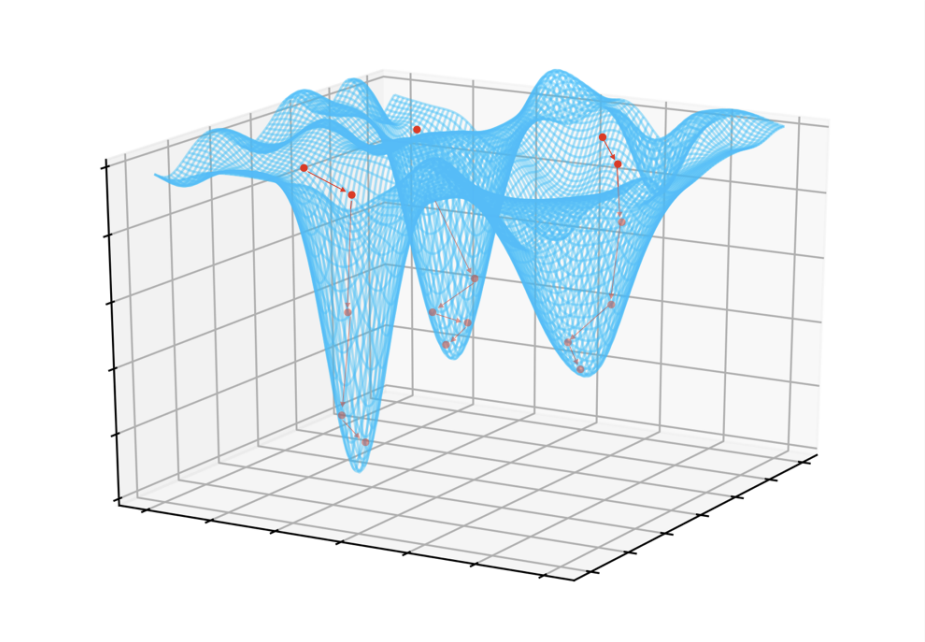


Figure 2.5: An illustration of deep ensembling.

Global Uncertainty and Mode Connectivity

Interestingly, the fictional loss surface depicted in Figure 2.5 is not representative of the full story. Recently, it has been found that the modes of a neural network's loss surface tend to be connected by (non-linear) pathways of low-loss [13], referred to as *loss tunnels*. Further, it is possible to traverse these loss tunnels so that we may produce an ensemble of global models without starting training from scratch each time, as demonstrated by *Fast Geometric Ensembling* (FGE) [20]. Note that this implies that local optima in the loss surface are not necessarily isolated, and thus the concepts of local and global uncertainty may lie on a more continuous spectrum than is suggested in idealised diagrams such as 2.5 or those in Section 1.1. While these loss tunnels are fascinating and important to mention, in practice we did not explore techniques such as FGE or the related Snapshot ensembling [29] as it was more convenient for our purposes to train ensembles members in parallel as per Algorithm 3.

Why do Deep Ensembles Work?

Deep Ensembles are quickly becoming a principal method for uncertainty representation in deep learning. They have repeatedly been shown to be empirically effective [41, 38, 49, 3]. However, although somewhat intuitive ("two heads are better than one"), it is not immediately clear why ensembling models should necessarily lead to better predictive performance, let alone more calibrated predictive uncertainty.

Often the benefit of ensembling is attributed to diverse predictions (or errors), and thus diverse ensembles of neural networks are often sought out [41]. One context in which there is a neatly packaged mathematical explanation for the effectiveness of ensembling and the benefit of diversity is when we desire to minimise the squared error. In this case, the *Ambiguity decomposition* [11] offers a satisfactory standalone explanation. Given a predictor $\bar{f} = \frac{1}{m}\sum_{i=1}^{m} f_i$ and some $(x, y) \in \mathcal{X} \times \mathbb{R}$, the Ambiguity decomposition is the identity:

$$(y - \bar{f}(x))^2 = \frac{1}{m}\sum_{i=1}^{m}(y - f_i(x))^2 - \underbrace{\frac{1}{m}\sum_{i=1}^{m}(f_i(x) - \bar{f}(x))^2}_{\text{ambiguity}}$$

Thus for some arbitrary example, the error of the ensemble is less than the average error of the individual predictors $(y-\bar{f}(x))^2 \leq \frac{1}{m}\sum_{i=1}^{m}(y-f_i(x))^2$. Furthermore, we can hypothetically push down the error of the ensemble by holding constant an individual predictor's accuracy while simultaneously increasing its deviation from the ensemble. Note that in the probabilistic setting, this allows us to neatly explain the effectiveness of ensembling in the limited context of (homoscedastic) Gaussian regression, since in this case minimising the squared error between predictions corresponds to minimising the negative logarithmic loss (see example 2.2.1), and thus ensembling diverse models leads to better predictions and better predictive uncertainty estimation.

General Setting

While the role of diversity in ensembling seems intuitive, in the general setting of probabilistic modelling, it is less well theoretically understood. While multiple decompositions exist explaining the general effectiveness of ensembling [33, 4], the explicit role of diversity and therefore a reasonable quantitative definition is not widely agreed upon, particularly regarding the negative logarithmic loss [64]. Despite this, empirical benefits of diversity are observable, and ensemble performance has been rigorously linked to the correlation of probability estimates under some assumptions [56], as well as in the classification setting [9, 10].

2.4.3 MultiSWAG

Bayesian Interpretation of Deep Ensembles

While deep ensembles were originally poised by Lakshminarayanan et al. [38] as an alternative to Bayesian methods for uncertainty estimation, Wilson et al. [61] point out that deep ensembles harmonise with the idea of approximating a Monte Carlo estimate of the Bayesian model average if we interpret the process of performing stochastic optimization as sampling from an approximate posterior. That is:

$$\begin{aligned}\mathbb{P}(y|x,\mathcal{D}) &= \int_{\theta\in\Theta} p_\theta(y|x)d\mathbb{P}(\theta|\mathcal{D}) \\ &:\approx \int_{\theta\in\Theta} p_\theta(y|x)dQ_{\text{opt}}(\theta|\mathcal{D}) \\ &\approx \frac{1}{m}\sum_{i=1}^{m} p_{\theta_i}(y|x) \quad \theta_i \sim Q_{\text{opt}}(\theta|\mathcal{D})\end{aligned}$$

where $Q_{\text{opt}}(\theta|\mathcal{D})$ is a distribution over parameter space induced by performing some variant of ERM on the loss surface defined by the training data $\mathcal{D}$, where the stochasticity comes from e.g. random initialisation or stochastic optimisation.

So long as the posterior concentrates increasingly on a finite set of parameters as $\mathcal{D}$ grows[4], then this "sampling via optimisation" procedure will come to approximate the posterior more and more closely, and the Monte Carlo estimate of the Bayesian model average will converge. Of course, with any practical amount of data, Q_{opt} is more or less guaranteed to vastly underweight the majority of parameter settings and by extension likely overweight the modes of the true posterior. Nevertheless, Wilson et al. view the effectiveness of deep ensembles as a compelling reason to investigate global uncertainty representation, leading to their proposal of MultiSWAG [61].

Local and Global Uncertainty

MultiSWAG combines the local uncertainty representation of SWAG with the global uncertainty representation of a deep ensemble by constructing multiple SWAG solutions to allow for approximate marginalisation over multiple different modes. In our implementation of MultiSWAG in Section 3, we keep the rank and model average samples assigned to each SWAG solution constant, and thus the algorithm for constructing a MultiSWAG solution is simply Algorithm 2 repeated with different initialisations in a manner analogous to Algorithm 3. Note that MultiSWA is the obvious analogy for the SWA solution.

[4]Note that due to the non-identifiability of parameter space, we cannot in full generality assume the posterior will collapse to a single point, in which case ironically an ensemble is the only way to compute the BMA.

Chapter 3

Experiments and Results

The first step in our experiments was to reimplement SWA [32], SWAG [43], Deep Ensembles [38], MultiSWA, and MultiSWAG [61]. We provide code for our implementations and the experiments that follow at:

`https://github.com/tmgrgg/localvsglobaluncertainty`

3.0.1 Method Complexities

Some of the experiments and discussions that follow explicitly talk about the relevant costs of the methods. To avoid repetition, Table 3.1 clarifies the relevant complexities for each method applied to a neural network with $|\theta|$ parameters. We tabulate space required to store the final trained model, and time required to predict using the final model.

Method	**Storage Space**	**Prediction Time**
SGD	$O(\|\theta\|)$	$O(\|\theta\|)$
SWA	$O(\|\theta\|)$	$O(\|\theta\|)$
MultiSGD/Deep Ens. (m SGDs)	$O(m\|\theta\|)$	$O(m\|\theta\|)$
MultiSWA (m SWAs)	$O(m\|\theta\|)$	$O(m\|\theta\|)$
SWAG (rank k, n samples)	$O(k\|\theta\|)$ / $O(n\|\theta\|)$	$O(kn\|\theta\|)$ / $O(n\|\theta\|)$
MultiSWAG (m SWAGs with (rank k, n samples))	$O(mk\|\theta\|)$ / $O(mn\|\theta\|)$	$O(mnk\|\theta\|)$ / $O(mn\|\theta\|)$

Table 3.1: Method Complexities

Note that the storage cost and prediction time of SWAG with rank k and n Bayesian model average samples depends on whether we store the k-rank approximation and sample n models at prediction time, or whether we presample

the models as part of the training process (and then discard the approximation used to generate them). The expressions on the left of the / in the SWAG row in Table 3.1 correspond to storing the approximation; the values on the right correspond to presampling. To be clear about where these complexities come from: if we do care about storing the rank k approximation itself, then this will cost $O(k|\theta|)$ as tabulated, and at prediction time we will need to actually sample models from the approximation. Each model can be sampled in $O(k|\theta|)$ time, resulting in an overhead of $O(kn|\theta|)$ for n models at prediction time. On the other hand, if we do not care about storing the SWAG approximation explicitly, then we can simply sample n models at the end of training and store these models, using them to predict as we would with any other ensemble. To avoid overcomplicating the discussions that follow, we will usually just talk about the minimum storage cost of SWAG $O(\min\{k|\theta|, n|\theta|\})$, and the minimum cost of prediction time with SWAG $O(n|\theta|)$. Note also that this nuance extends to MultiSWAG which we shall handle similarly.

3.0.2 Architectures and Datasets

In our experiments in Section 3, we will predominantly focus on the setting of multiclass image classification, meaning we will work with datasets consisting of images belonging to exactly one out of several fixed possible classes.

Dataset: CIFAR-10

The CIFAR-10 [37] dataset consists of 50,000 training images and 10,000 test images. There are 6000 coloured images for each of the 10 different classes: aeroplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each image is 32x32.

Dataset: FashionMNIST

Fashion-MNIST is a dataset of images of clothing items, with a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image depicting an item from one of 10 classes: t-shirts, trousers, jumpers, dresses, coats, sandals, shirts, trainers, bags, and ankle boots.

Models

We will work with various forms of convolutional architecture. We construct two Pre-activation Residual Networks [27], one with 20 layers (PreResNet20), and the other with 164 layers PreResNet (PreResNet164). We will also work with a (small) Densely Connected Convolutional neural network [30] with 10 layers (DenseNet10). Finally, to represent an older architecture, we will work with LeNet5 [40]. We train all models with the default hyperparameters and regularisation used in [43] to train the PreResNet164 in Maddox et al. [43].

3.0.3 Metrics

Suppose we have a probabilistic classifier $p : \mathcal{X} \to (\mathcal{Y} \to \mathbb{R})$, where $\mathcal{Y} = \{c_1, c_2, \ldots, c_k\}$ for k classes. Suppose also we have a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$. The primary metrics we will discuss in our experiments are as follows:

Accuracy

$$Accuracy(p, \mathcal{D}) = \frac{1}{m} \sum_{i=1}^{m} \left[\left(\operatorname*{argmax}_{y \in \mathcal{Y}} p(y|x) \right) = y_i \right]$$

where [`cond`] is 1 if the condition is true and 0 otherwise.

Negative Logarithmic Loss (NLL)

$$NLL(p, \mathcal{D}) = -\sum_{i=1}^{m} \log(p(y_i|x_i))$$

(Expected) Kullback-Leibler Divergence

Suppose we have another probabilistic classifier $q : \mathcal{X} \to (\mathcal{Y} \to \mathbb{R})$, then:

$$KL(p|q) = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{y \in \mathcal{Y}} p(y|x_i) \log \left(\frac{p(y|x_i)}{q(y|x_i)} \right) \right)$$

Reliability Diagram

Note that we will use a modified version of the reliability diagrams presented in 2.2. Essentially, we will rotate them so that the x-axis corresponds to perfect calibration. This is achieved by plotting model confidence minus model accuracy on the y-axis, and confidence on the x-axis.

Expected Calibration Error

Our implementation of expected calibration error is adaptive, and so despite being conceptually simple, in the interest of brevity we refer to the `calibration_curve` function in the repository.

3.1 Performance as Individual Solutions

We begin with a direct comparison of SGD, SWA, and SWAG models. This allows us to establish the relative performances of each method, and to empirically confirm the benefit of representing local uncertainty. In the below, we use a rank 30 SWAG solution with 30 local samples.

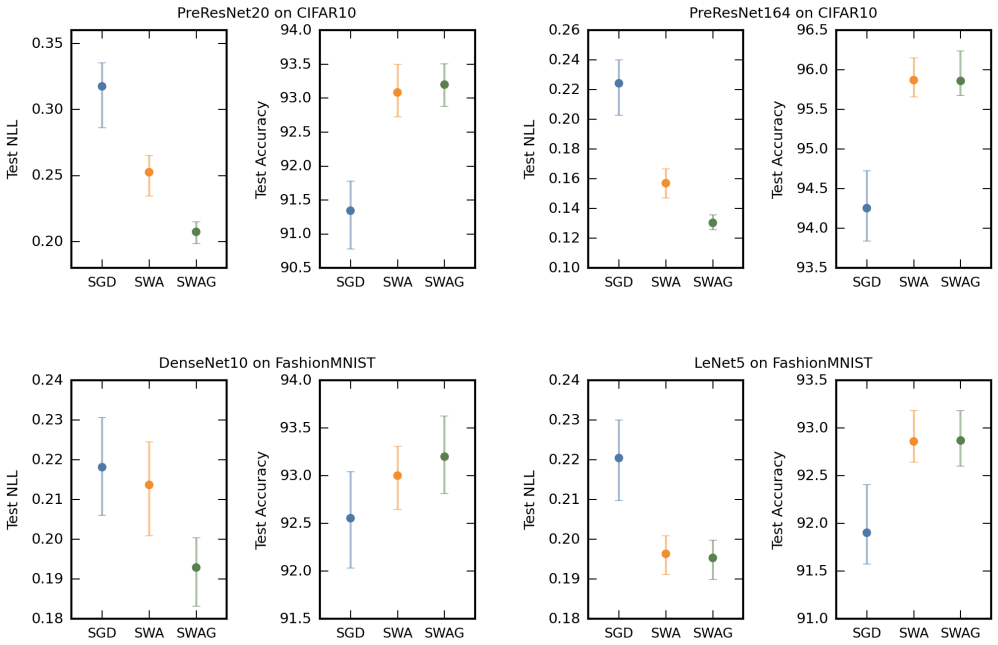


Figure 3.1: Negative logarithmic loss (NLL) and accuracy evaluated on test data for SGD, SWA, and SWAG solutions.

NLL and Accuracy

From Figure 3.1 it is clear that the solutions are consistently ranked in order of asscending performance as SGD, SWA, and SWAG, corroborating the results of [43]. We note that SWA's performance seems to deviate between the performance of SGD and SWAG depending on the model and dataset.

Calibration

As discussed in Section 2.1, good predictive uncertainty should lead to a well-calibrated model. As such, in Figure 3.2 we plot reliability diagrams for the SGD, SWA, and SWAG solutions on each model-dataset pair. Note that points above the x-axis correspond to regions where the models confidence exceeds the model's accuracy (overconfidence), whereas points below correspond to regions where the model is more accurate than it is confident (underconfidence).

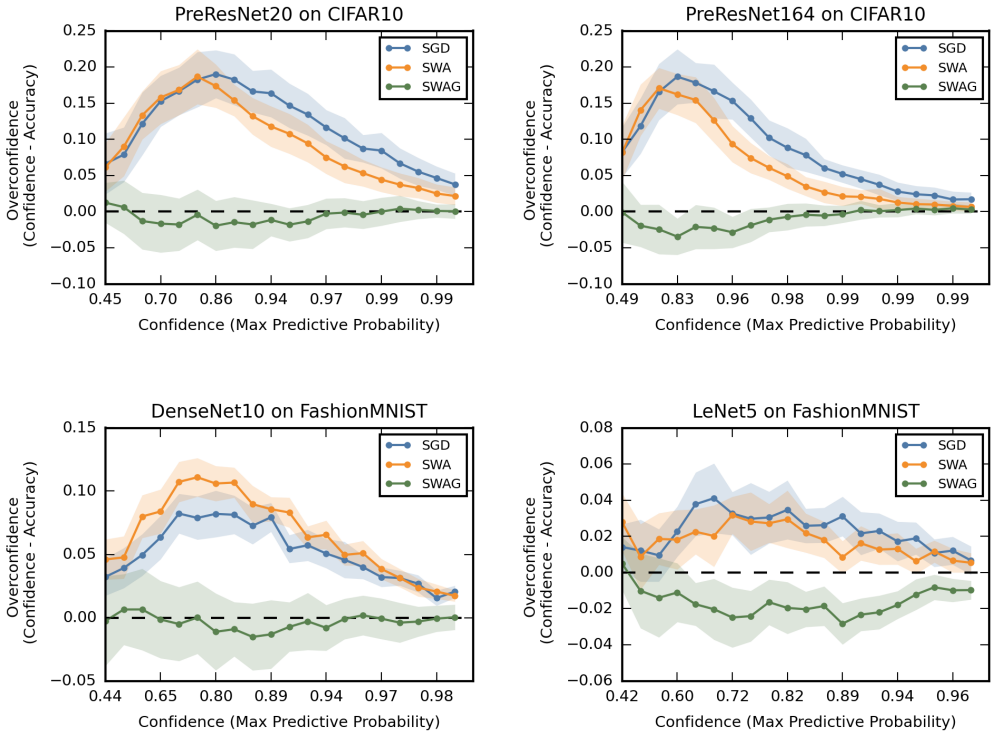


Figure 3.2: (Averaged) Reliability diagrams for SGD, SWA, and SWAG.

From Figure 3.2 it is clear that the SGD solution is consistently overconfident across all regions of model confidence, as suggested by [24]. SWA is also consistently overconfident, though comparatively less so than SGD, perhaps owing to its increased predictive accuracy. SWAG appears to be well-calibrated, with a slight tendency towards underconfidence, especially in the older LeNet5 model.

Remarks

Our experiments confirm that both SWA and SWAG lead to improvements in predictive performance over standard a standard SGD solution. Additionally, SWAG admits far better calibrated predictive uncertainty when compared to either SGD or SWA. Note however that a rank k SWAG approximation represents k times the storage cost of either the SGD or SWA solutions, and requires m model samples to be drawn from its representation of local uncertainty in order to compute predictions. In the above experiments we set $k = m = 30$, meaning the SWAG solution is at least 30 times more expensive to store and to predict with. Reversing Wilson et al.'s [61] perspective that deep ensembles are Bayesian, equivalenty, we could consider SWAG as an ensemble of m models which happen to be drawn from the same local subspace. This raises the question: how does SWAG compare to a traditional deep ensemble?

3.2 Comparison as Ensembling Techniques

Here, we explore SWAG's performance as an ensemble against both a traditional deep ensemble and an ensemble of SWA solutions (MultiSWA). Note that the storage and computation cost of each of these methods scales identically with the number of ensemble members. For deep ensembles and MultiSWA, each additional model in the ensemble is trained with identical hyperparameters but a distinct initialisation. For SWAG, each additional model is sampled from the constructed Gaussian approximation.

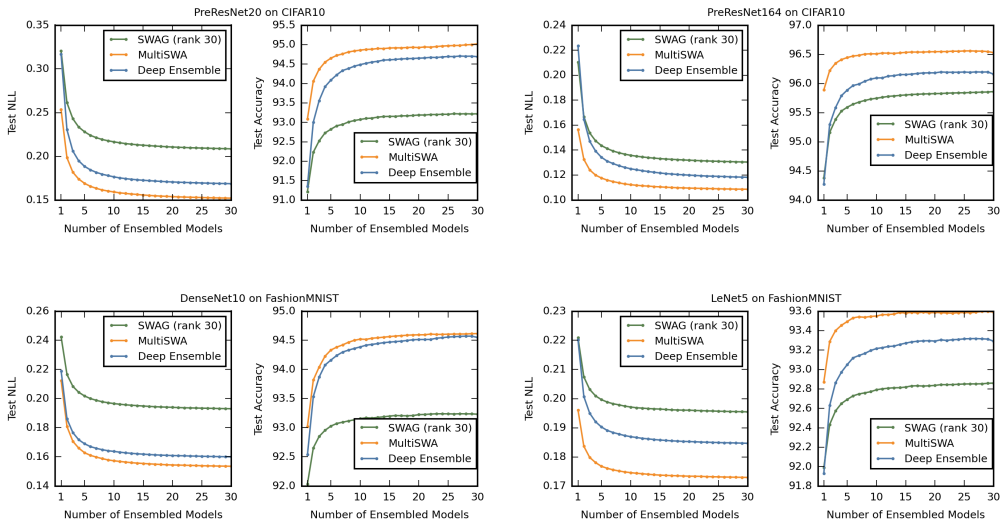


Figure 3.3: NLL and Accuracy of Deep Ensemble, MultiSWA, and SWAG solutions plotted as a function of the number of ensemble members. Results averaged over 10 runs.

NLL and Accuracy

Figure 3.3 shows clearly that, in terms of both NLL and accuracy, a deep ensemble of 30 "global model samples" substantially outperforms SWAG with 30 local model samples. Further, a relatively small deep ensemble of about 2 or 3 models achieves similar or better predictive performance to a SWAG solution consisting of 30 models. MultiSWA has even better predictive performance, consistently outperforming a traditional deep ensemble for all ensemble sizes. These results clearly suggest that, at least in terms of NLL and accuracy, ensembling global solutions leads to better performance per unit storage and time than locally sampling from SWAG.

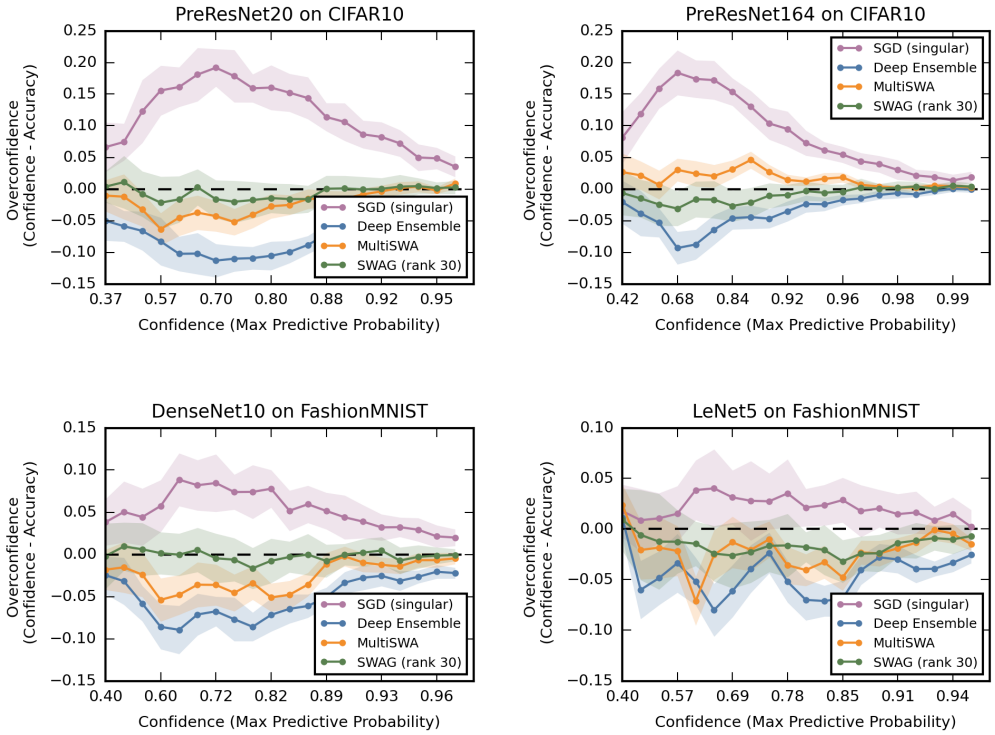


Figure 3.4: Reliability Diagrams for SWAG (rank 30, 30 models), SGD Ensemble (30 models), MultiSWA (30 models).

Calibration

We saw in Section 3.1 that SWAG produces well-calibrated predictions relative to a singular SGD or SWA solution. Traditional ensembling is also widely believed to lead to better calibrated models [38, 41]. To compare the approaches, in Figure 3.4 we plot the reliability curves for each of the SWAG, Deep Ensemble, and MultiSWAG ensembling techniques. It is clear from Figure 3.4 that all three methods correct for the overconfidence of a singular solution. Both Deep Ensembles and MultiSWA appear to overcorrect to a substantial degree. SWAG appears to remain the most well-calibrated solution. Recall from Section 2.1 that while NLL is technically a measure of calibration, it is particularly insensitive to both overconfidence and underconfidence in regions of high accuracy, and therefore the reliability diagram is a more direct (and interpretable) representation of the model's calibration.

3.2.1 Underconfidence of Deep Ensembles

Interestingly, it appears that as the number of models in a Deep Ensemble grows, the model becomes increasingly underconfident, as can be seen in the left hand plots in Figure 3.5. We did not observe this effect with SWAG, whose calibration appeared to increase monotonically with the number of ensembled models. The graphs on the right of Figure 3.5 demonstrate this discrepancy concisely by plotting the ECE [24] as a function of the number of models.

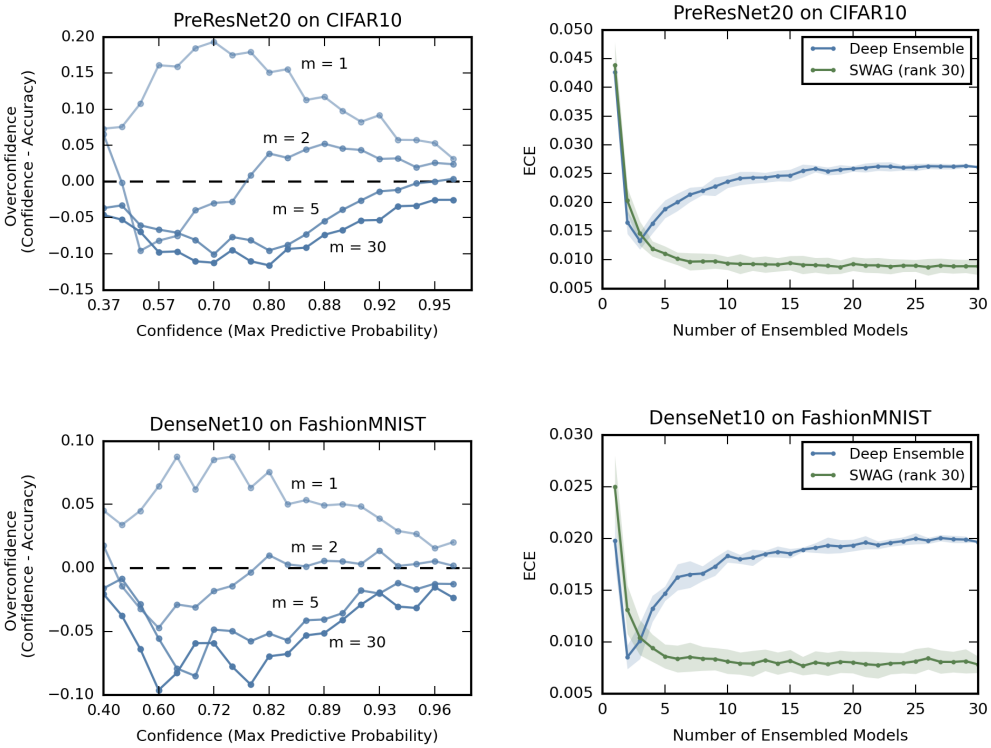


Figure 3.5: Left: Reliability curves plotted for Deep Ensemble of size $m = 1$, $m = 2$, $m = 5$, $m = 30$, demonstrating a sharp transition from overconfidence to under-confidence. Right: Expected Calibration Error (ECE) for Deep Ensembles and SWAG as number of models increases.

Remarks

While superior predictive performance of deep ensembles over SWAG is indisputable, our experiments seem to suggest that focusing purely on global uncertainty tends to lead to miscalibration in the form of underconfidence, despite monotonic improvements in NLL. This corroborates similar findings in [52, 60]. While this overcorrection is typically overall beneficial due to the overconfidence of individual neural network models trained via SGD, the procedure of ensembling multiple global models itself does not automatically lead to better calibration.

3.3 Analysis of SWA-Gaussian's Local Subspace

So far we have fixed the hyperparameters of SWAG's local subspace. However, getting an idea for the properties of the space from which SWAG draws it model samples is crucial for understanding SWAG's flavour of local uncerainty. As such, here we investigate this space.

3.3.1 Performance versus Subspace Rank

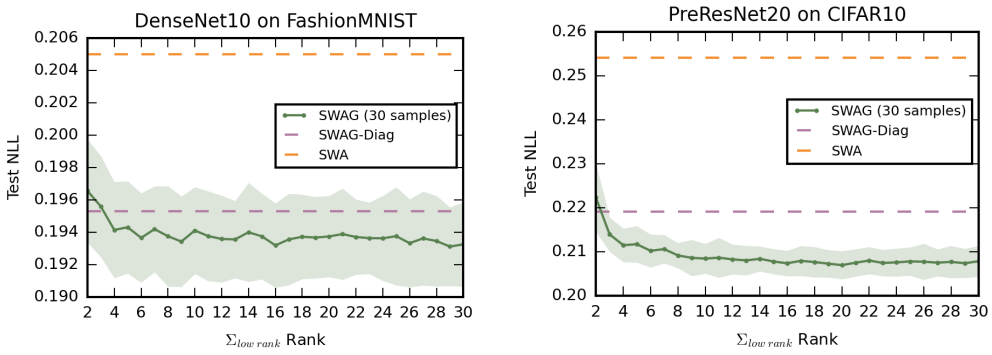


Figure 3.6: SWAG NLL as approximation rank increases. The performance of corresponding SWA solution and SWAG-Diag solution are plotted as dashed lines.

NLL

We directly assess the effect of approximation rank on performance in terms of NLL in Figure 3.6. Empirically we find a sharp improvement in NLL as approximation rank increases initially. The benefit of increasing rank quickly diminishes, with ranks beyond roughly 15 in both of the above plots demonstrating little to no clear improvement. This is noteworthy since the storage space cost of the full approximation is proportional to its rank, and thus storage may be saved in practice by using a lower rank approximation.

Saturation

Figure 3.6 gives no information about the relationship between the approximation rank and the number of models samples in the solution; in particular it seems plausible that SWAG solutions of higher rank may require more model samples to fully "saturate" their model average over a higher dimensional/more voluminous

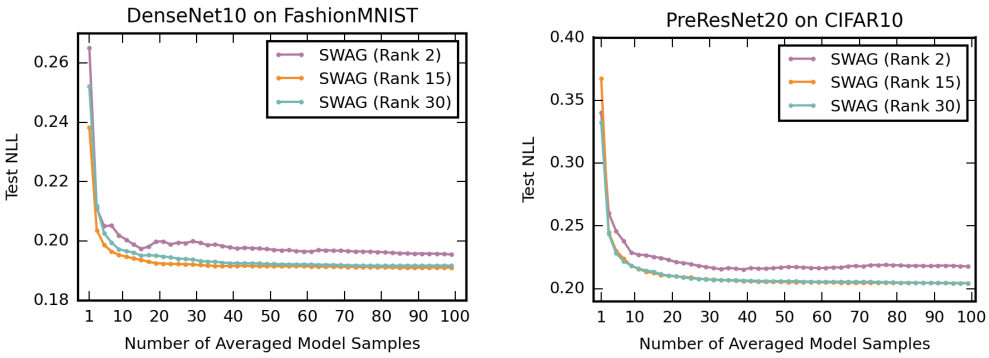


Figure 3.7: NLL for SWAG approximations of ranks 2, 15, and 30 as the number of models in the model average increases.

subspace. Here, we (loosely) define a model average/ensemble to be *saturated* once it stops improving with the addition of more models. To explore this, in Figure 3.7 we plot the performance (in terms of NLL) for SWAG solutions with different ranks 2, 10, and 30 as we increase the number of model samples in the model average up to 100. The plots suggest that the rank 15 and rank 30 SWAG solutions saturate at roughly the same number of model samples. Furthermore, there appears to be very little difference in performance between the saturated model averages for ranks 15 and 30. This confirms that the performance tail-off of higher ranks observed in the previous Figure 3.6 is not an artefact of reduced saturation due to the fixed number of model samples.

Rank and Number of Models

For completeness, in Figure 3.8 we additionally plot heatmaps comparing the performance of SWAG models with variable rank and variable number of local model samples included in the model average. From these, it is clear that increasing the number of local model samples and increasing the rank of the approximation leads to complementary performance benefits. The benefit of increasing rank appears to tail-off in accordance with Figure 3.6. The gradient transition from left to right is much more gradual than the gradient transition from bottom to top, meaning the number of model samples included in the model average is arguably more critical than the rank of the approximation, particularly after the first few ranks. This is unsurprising when considering SWAG as an ensemble, since it is generally common for one (or a few) "very good" models to be outmatched by an ensemble of many "reasonably good" models. Note that the minimum storage cost of the solution at a given square in the heatmap is proportional to the minimum of its row and column position as per Table 3.1.

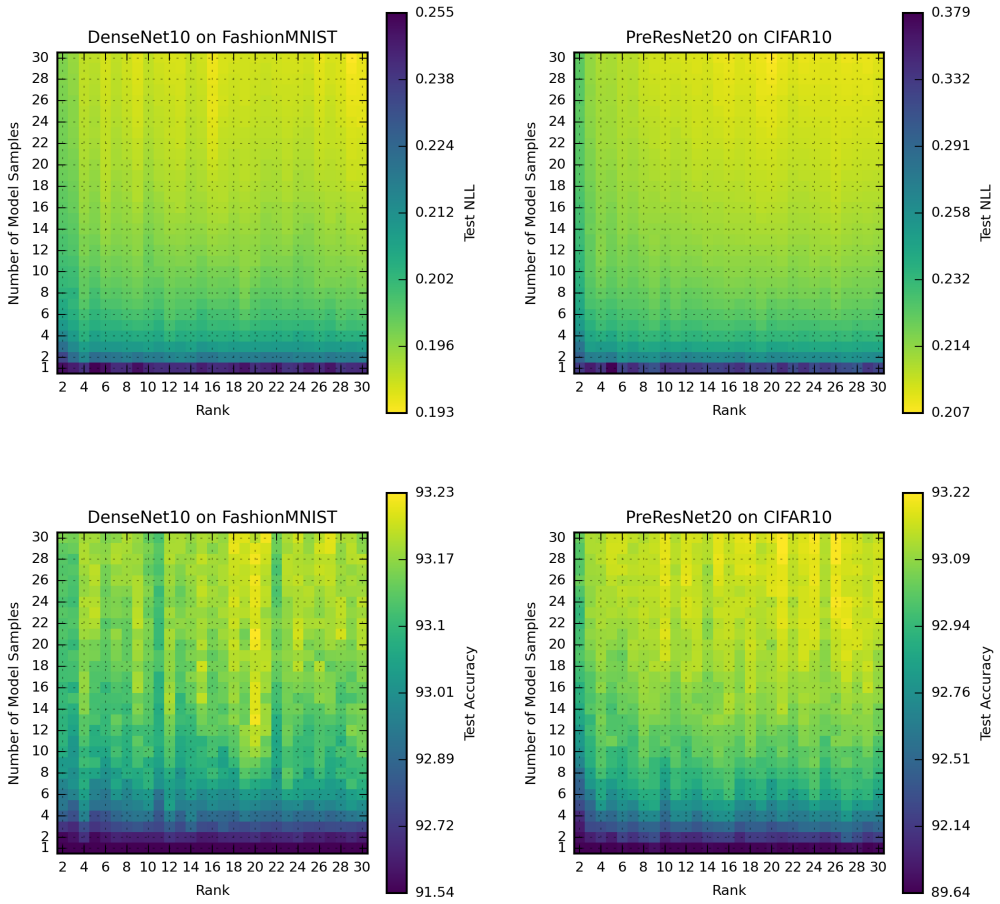


Figure 3.8: Top row: Heatmap showing NLL for model with rank x and number of local samples y. Bottom row: Accuracy represented similarly. Results smoothed by averaging experiments.

3.3.2 Covariance and Scale

The results of the previous experiments suggest a significant tail-off of performance benefits for increasing the approximation rank, and that this tail-off does not appear to have to do with decreased saturation/coverage of the represented distribution for a fixed number of samples. In order to get an idea for the volume of the space represented by SWAG, in Figure 3.9 we plot the magnitude of the singular values of a rank 30 $\Sigma_{\text{low rank}}$ matrix, as well as the magnitude of the largest 30 diagonal elements of Σ_{diag}

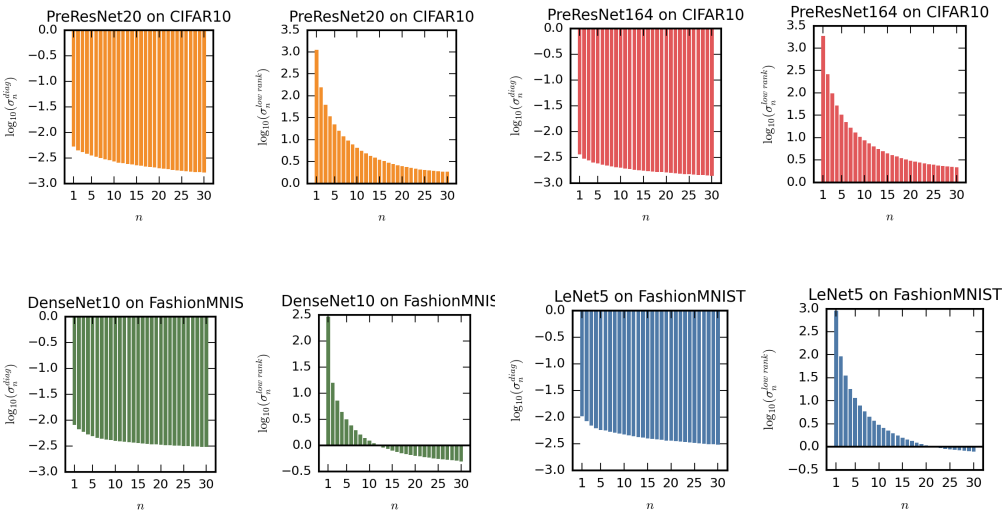


Figure 3.9: Left column: we plot the logarithm (base 10) of the singular values of $\Sigma_{\text{low rank}}$ for a rank 30 SWAG approximation in descending order. Right column: we plot the logarithms of the 30 largest singular values of Σ_{diag}.

It is clear from Figure 3.9 that the covariance of network parameters within the low-rank subspace captured by $\Sigma_{\text{low rank}}$ is dominated by a small number of high variance directions. This perhaps explains why we see sharply diminishing returns for increasing the rank of the SWAG approximation beyond a certain threshold in Section 3.3.1. Loosely speaking, as the number of ranks increase, their effect on the size of the space which we are sampling from is diminishing, i.e. each additional dimension admits smaller and smaller movement in a new direction. This also explains why a rank 15 space takes similar (potentially fewer) samples to saturate than a rank 30 space: the samples essentially have to cover the same "important region" in parameter space. Notably, we might therefore expect lower ranks to actually saturate with fewer samples since there is less 'noise' volume in the lower dimensional space, and in fact we do observe this in the DenseNet plot in Figure 3.7.

Also noteworthy is that the directions of greatest variance within the low-rank subspace are several magnitudes greater than those in the diagonal. This is perhaps to be expected, as it is unlikely that the directions of greatest variance in the posterior/loss-surface would be axis-aligned. This calls in to question the role of the diagonal matrix.

3.3.3 How Important is the Diagonal?

Assuming we stay within a local mode, intuitively we might expect the correlations between weights to be more important than the weights themselves, given that the scale of any numerical feature representation is somewhat arbitrary. Furthermore, the variance information is also contained in the diagonal of the low-rank matrix, albeit not independently of other weights. This variance in axis-aligned directions was also shown to be small in our previous experiments. Maddox et al. [43] investigated SWAG-Diag, but did not investigate the omission of this diagonal matrix. It is therefore unclear what effect, if any, the diagonal matrix has on the approximation.

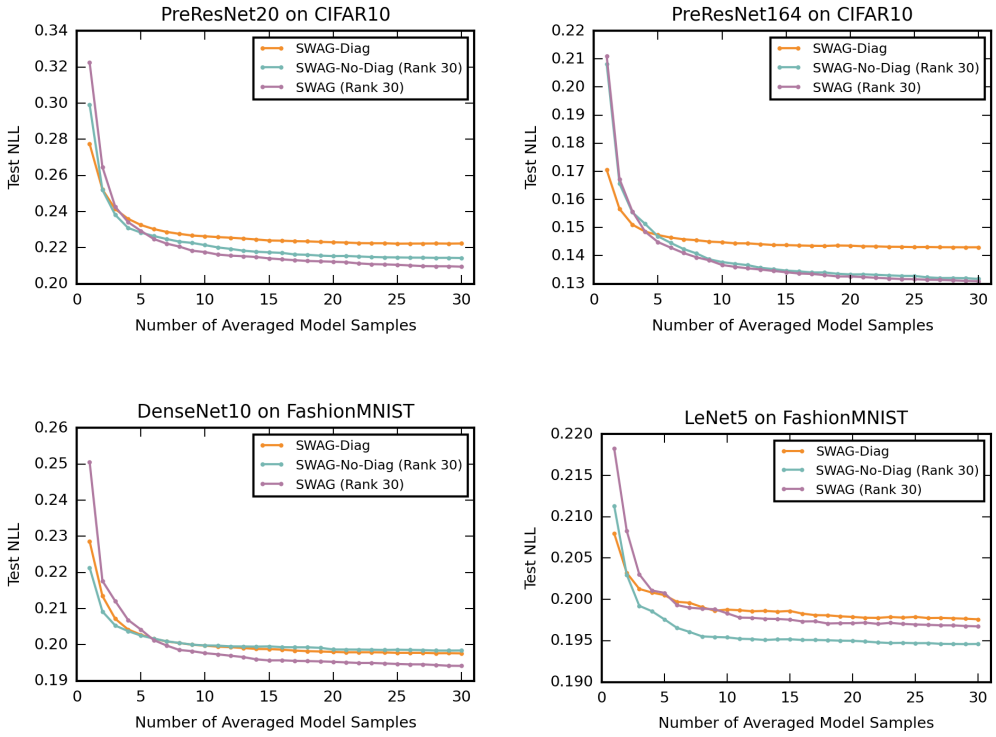


Figure 3.10: NLL of SWAG, SWAG-Diag, SWAG-No-Diag with covariance $\frac{1}{2}(\Sigma_{\text{diag}} + \Sigma_{\text{low rank}})$, Σ_{diag}, $\Sigma_{\text{low rank}}$, respectively.

In 3.10 we plot the NLL for increasing model samples of SWAG with only the low-rank matrix (i.e. SWAG-No-Diag), as well as SWAG-Diag and SWAG as originally proposed by Maddox et al. [43]. Interestingly, we observe largely inconsistent effects. This suggests that we cannot generally be certain that removing the diagonal matrix will not negatively affect the quality of the final SWAG solution.

Scale Parameter

An important hyperparameter of SWAG is the factor by which we scale the covariance. From an approximate Bayesian perspective, this hyperparameter can be thought of as a variational parameter; we would expect the SWAG solution to perform better for whichever value of the parameter minimises the difference between the otherwise fixed local Gaussian approximation and the posterior. Equivalently, we may expect the ensemble formed by SWAG to have different properties as we draw models farther from the centre of our approximation. In Figure 3.11 we plot the NLL of SWAG, SWAG-No-Diag, and SWAG-Diag using different scale parameters.

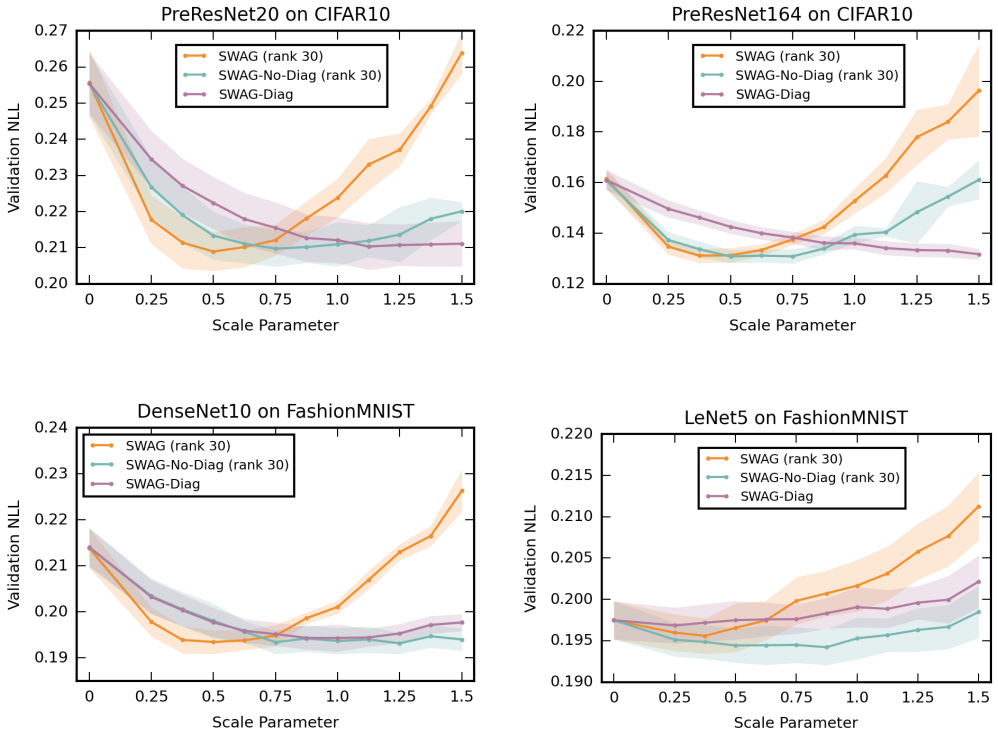


Figure 3.11: Scale parameter value plotted against resulting validation NLL for a rank 30 SWAG solution with 30 model samples. Note that the SWA solution (i.e. SWAG mean) corresponds to the scale parameter being 0.

As reported in [43], the optimal scale parameter for SWAG tends to be close to 0.5, though not exactly. It is interesting to see that the optimal scale parameter for the SWAG-No-Diag solution is variable between model-dataset pairs, often being below 1.0. By contrast, SWAG-Diag appears to suffer if a scale parameter less than 1 is used, and actually benefits to a small degree from a scale larger than 1. This altogether suggests that perhaps we should consider using separate scale parameters for each matrix, though we shall avoid doing so in the experiments that follow for the sake of consistency.

3.4 Analysis of Diversity

In 3.2, we saw that while SWAG was arguably better calibrated than a deep ensemble, its predictive performance was substantially worse. In practice, this trade-off may make it difficult to justify choosing a SWAG solution over a traditional ensemble, especially given their equivalence in cost. Referring to our discussion in Section 2.4.2, this is somewhat unsurprising, as diversity is intuited to play a significant part in ensemble performance. This relates to global and local uncertainty because we might expect models contained within a local subspace to be less diverse than models drawn from different modes across the loss landscape.

Note that the idea that closeness in parameter space implies closeness in prediction space is nuanced in general. While the functional represented by a neural network is differentiable and therefore continuous, in practice we do not know the nature of this continuity. That is, without making further regularity assumptions, we cannot in general bound the difference that moving some small distance in parameter space will make to the output of the corresponding predictive functional.

Despite this, there is empirical evidence to support that at least in practice closeness in parameter space implies a higher probability for closeness in predictive space. Fort et al. [17] empirically demonstrated that the diversity of the hard predictions made by some classes of neural networks was higher in models across different modes versus models from a local subspace. The overarching purpose of the experiments in this section is therefore to empirically investigate the relationship that diversity has with the flavour of local and global uncertainty represented by SWAG and deep ensembles.

3.4.1 Pairwise Diversity of Methods

As mentioned in Section 2.3.2, the Kullback-Leibler divergence describes a natural notion of discrepancy between two distributions. Here, we shall use this as a raw diversity score. In Figure 3.12 we plot the sample densities of the observed KL divergences between pairs of predictive distributions arrived at via SGD, SWA, and SWAG considered as sampling procedures. These divergences were computed on and averaged over a validation set. To be clear: here we sample both distributions from the same SWAG approximation, and arrive at different SGD solutions and SWA solutions by stochastic gradient descent.

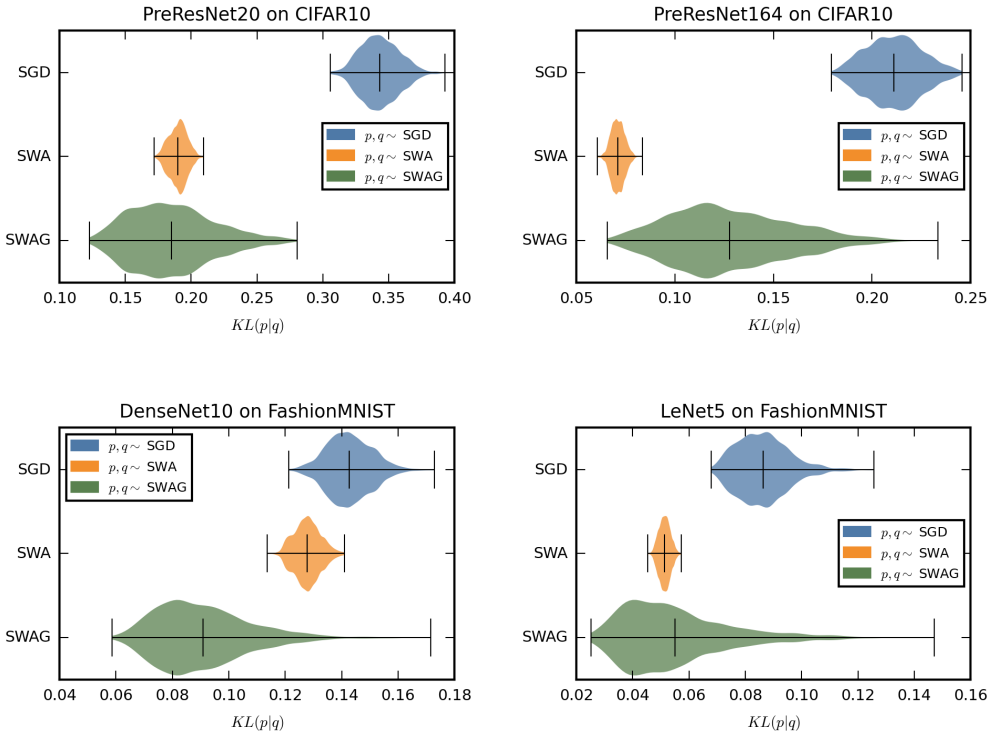


Figure 3.12: KL divergence densities observed between pairs of predictive distributions arrived at via SGD, SWA, and by *sampling* from a given SWAG solution.

Figure 3.12 demonstrates that the pairwise KL diversity is generally lower between pairs of SWAG samples than it is between pairs of SGD solutions. This supports the idea that models arrived at by SGD tend to be more diverse than those sampled from a local SWAG approximation. Interestingly, the pairwise diversity score of the SWA solution is highly model-dataset dependent. In 3 out of 4 cases, the mean KL divergence of SWA is similar to or lower than that of SWAG. In all cases the KL divergence is visibly lower for SWA than it is for SGD. Note that MultiSWA had substantially better predictive performance than both SWAG and deep ensembles in Section 3.2. The lower pairwise diversity of MultiSWA does not contradict the idea that diversity is a factor in the relative performance of these ensembling techniques, as an individual SWA solution also has better predictive performance than an SGD solution or a single SWAG model sample, and thus likely requires less diversity in order to benefit from ensembling.

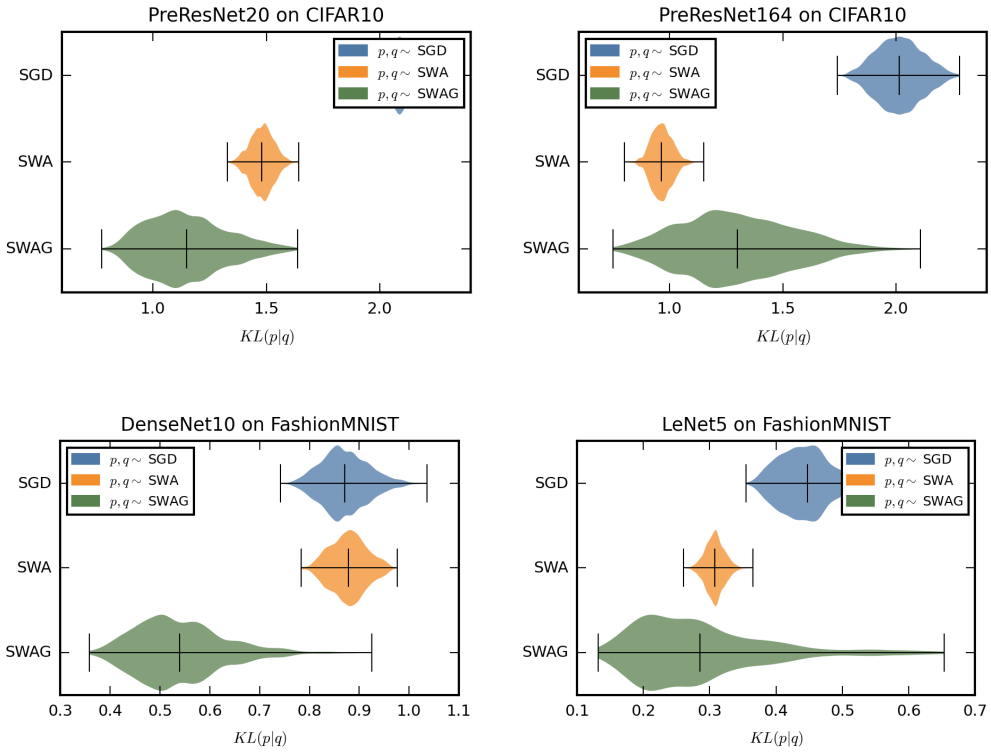


Figure 3.13: KL divergence densities as in Figure 3.12, but here only looking at examples in the validation set where at least one of the solution pair would make an incorrect prediction.

Accounting for Predictive Performance

One possibile confounding factor is the predictive performance of the solutions. Models that have superior predictive performance are naturally going to have less predictive diversity because they will more often agree on the correct answer, by nature of being correct more often. In Figure 3.13 we account for this by only computing the KL divergence on examples in the validation set where at least one of the pair of solutions is incorrect.

We observe that although the KL divergences increase overall in magnitude from Figure 3.12 to Figure 3.13, the relative pairwise diversity scores of the different solutions change only subtly. The SWA error diversity remains lower than or similar to SGD in all cases. For PreResNet164 the SWA error diversity is still significantly lower than SWAG. This suggests that the predictive performance of MultiSWA could potentially be improved if we could hold the predictive performance of each solution constant while somehow increasing diversity.

3.4.2 Local versus Global Diversity

Our original intuition was that globally sampled models (i.e. SGD, SWA) are more diverse than locally sampled models from a given SWAG approximation. While we may surmise from Figures 3.12 and 3.13 that there is some truth to this, the observation is somewhat muddied by the notion that SWA models can have relatively low diversity despite theoretically corresponding to "centralised" solutions in *different* modes/basins of attraction. To get a more direct perspective purely on the matter of local versus global diversity, in Figure 3.14 we plot the KL-divergences of pairs of samples from the same SWAG approximation, versus pairs of samples sampled from different SWAG approximations.

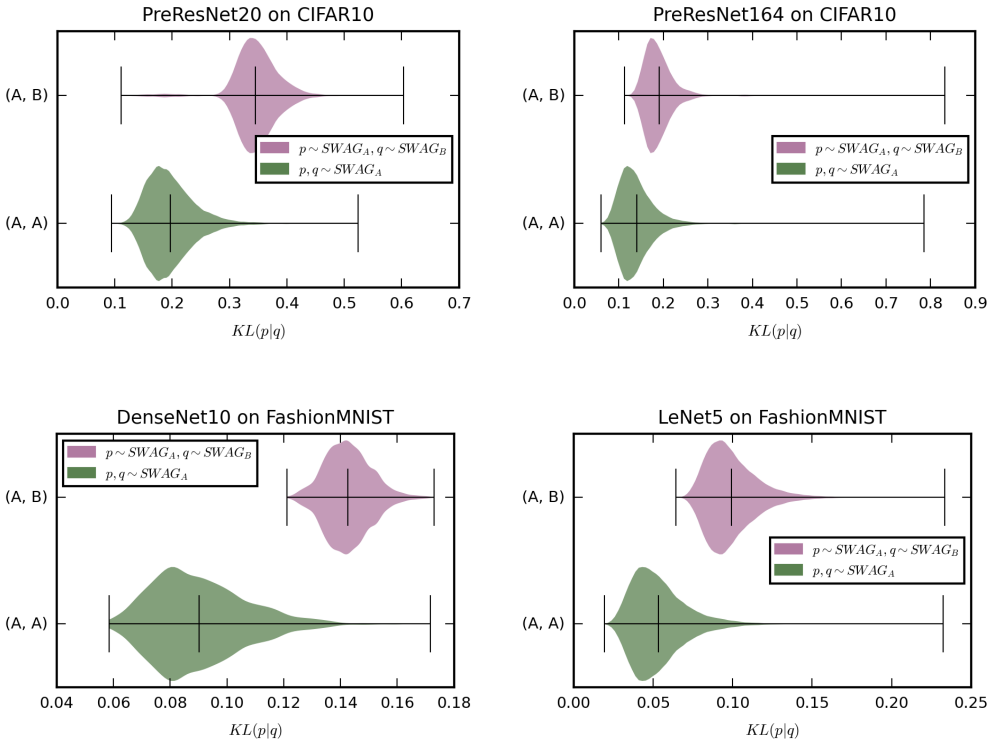


Figure 3.14: Violin plots illustrating the difference in densities between Kullback-Leibler divergences of models sampled from the same SWAG approximation as compared to models sampled from different SWAG approximations.

Figure 3.14 demonstrates that if we sample from SWAG intramodally then we see less diversity than we do if we sample from SWAG intermodally. This suggests that generally we can expect models sampled from a single mode to be less diverse than globally sampled models. Since the SWA solution is the centre of the SWAG solution, this suggests that while different basins of attraction correspond somewhat to different predictive solutions, this diversity decreases as we move more towards the centre.

3.5 Representing Both Local and Global Uncertainty

So far we have kept the ideas of local and global uncertainty separate from one another, i.e. examining the former through SWAG and the latter through deep ensembles/MultiSWA. The natural extension of this as proposed by Wilson et al. [61] is to consider MultiSWAG, wherein we ensemble SWAG solutions so as to represent both local and global uncertainty. In Figure 3.15 we plot both NLL and accuracy as a function of the number of ensembled models for a traditional deep ensemble, MultiSWA, and a MultiSWAG solution where each SWAG approximation is rank 30 and uses 30 local model samples. Note that in contrast to previous experiments, in Figure 3.15 we treat an entire SWAG approximation as one additional model, despite the fact that it consists of 30 local model samples.

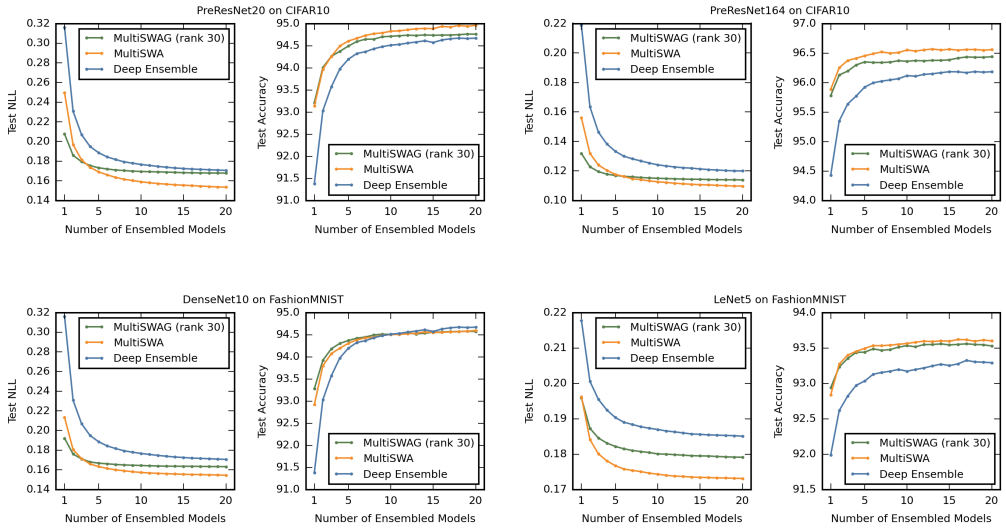


Figure 3.15: NLL and Accuracy of deep ensemble, MultiSWA, and MultiSWAG solutions plotted as a function of the number of ensemble members. Results averaged over 5 runs

It is somewhat counterintuitive to see that the predictive performance of MultiSWAG consistently ranks below that of MultiSWA. Firstly, from the perspective of the performance of the individual solutions: SWA is significantly outperformed by the SWAG solution in Section 3.1. Secondly, from the sheer difference in resources: the ensemble corresponding to the MultiSWAG solution is 30 times larger than that of the deep ensemble and MultiSWA, i.e. 20 SWAG models corresponds to 600 parameter vectors, one for each local model sample.

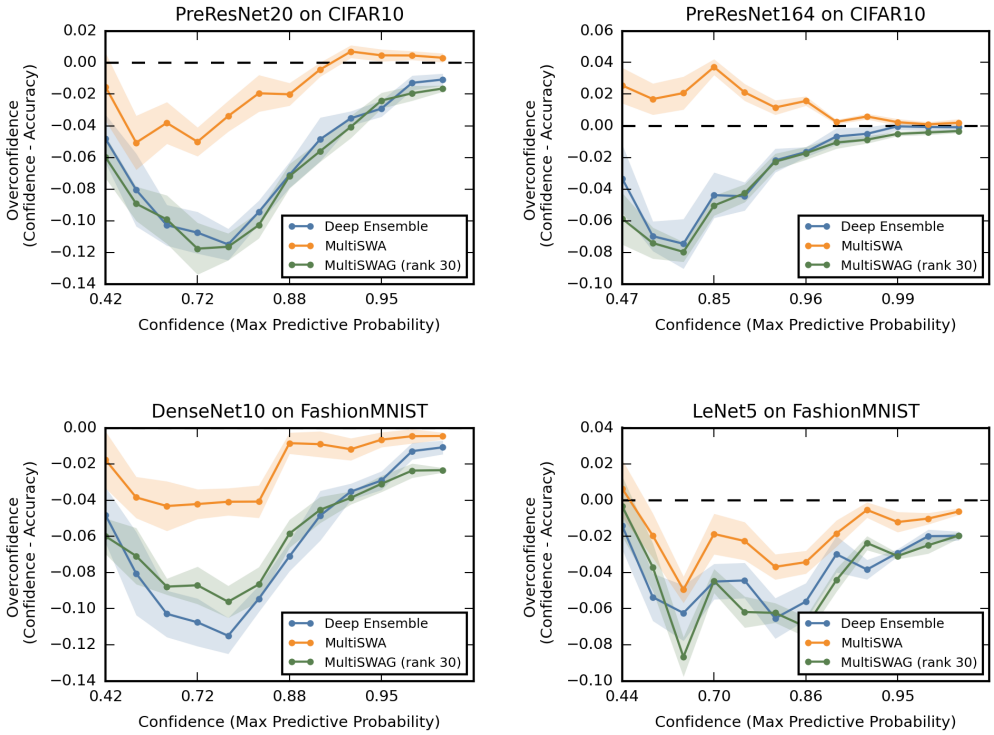


Figure 3.16: Reliability Diagrams for MultiSWAG (20 models, each rank 30 with 30 local model samples), SGD Ensemble (20 models), MultiSWA (20 models).

In Figure 3.16 we plot reliability diagrams for the deep ensemble, MultiSWA, and MultiSWAG solution. We find that the MultiSWAG solution appears to suffer from the same issues with underconfidence as deep ensembles did in Section 3.2.1. This is also somewhat surprising, given that the individual SWAG solutions were well-calibrated compared to MultiSWA and deep ensembles. In Figure 3.17 we observe the extreme nature of this effect: the expected calibration error *increases monotonically* with the number of SWAG solutions we ensemble!

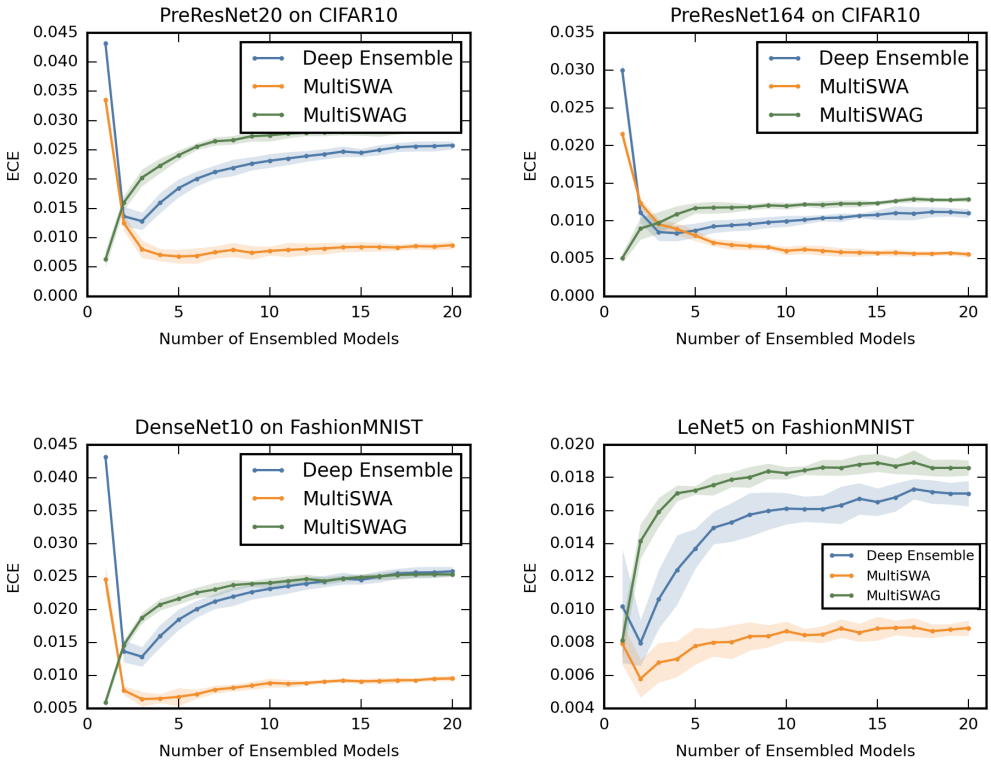


Figure 3.17: Expected Calibration Error of deep ensemble, MultiSWA and MultiSWAG as a function of number of ensembled models.

Figure 3.17 demonstrates that ensembling SWAG solutions suffers from the same underconfidence effect that ensembling SGD solutions suffered in Section 3.2. MultiSWA suffers similarly in 2 out of the 4 model-dataset pairs, though to a substantially lesser extent. For MultiSWAG, this is especially counterintuitive because in Section 3.2 we observed that the Expected Calibration Error decreased monotonically as the number of local model samples grew. It therefore appears to be the case that ECE tends to grow (due to increased underconfidence) for all model types as we increase the number of global model samples. This suggests there may be some trade-off to the number of local and global models, motivating a multidimensional analysis of MultiSWAG. In Figure 3.18 we plot the performance of a MultiSWAG solution as we increase the number of local model samples as well as the number of (rank 30) SWAG solutions included in the model, for both of the PreResNet models on CIFAR10. We similarly plot the Expected Calibration Error.

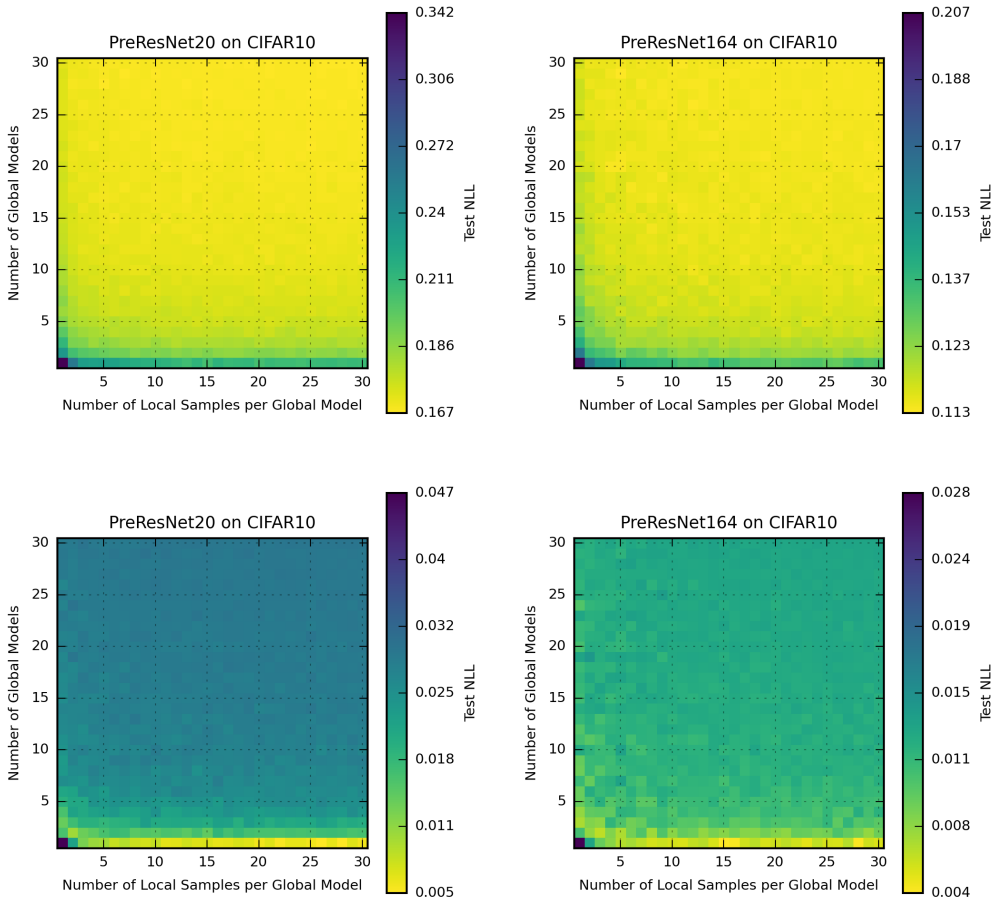


Figure 3.18: Top row: NLL for a MultiSWAG model using [row] global models each with [column] local samples. Bottom row: Similarly for ECE.

Figure 3.18 demonstrates a conflict between expected calibration error and NLL. According to the heatmaps in the top row, adding local models beyond the first few has minimal effect on reducing loss. On the other hand, the bottom row tells us that adding global models actually increases the expected calibration error, and only by keeping one global model and increasing the number of local samples can we expected reasonable calibration. Thus, the two concepts of local and global uncertainty as represented by MultiSWAG seem to be at odds with one another, with global uncertainty improving predictive performance and sacrificing calibration, and local uncertainty providing good calibration but comparatively poor predictive performance. We further plot 3.19 to explicitly demonstrate the negligible effect of increasing our local uncertainty representation on the NLL of a MultiSWAG solution.

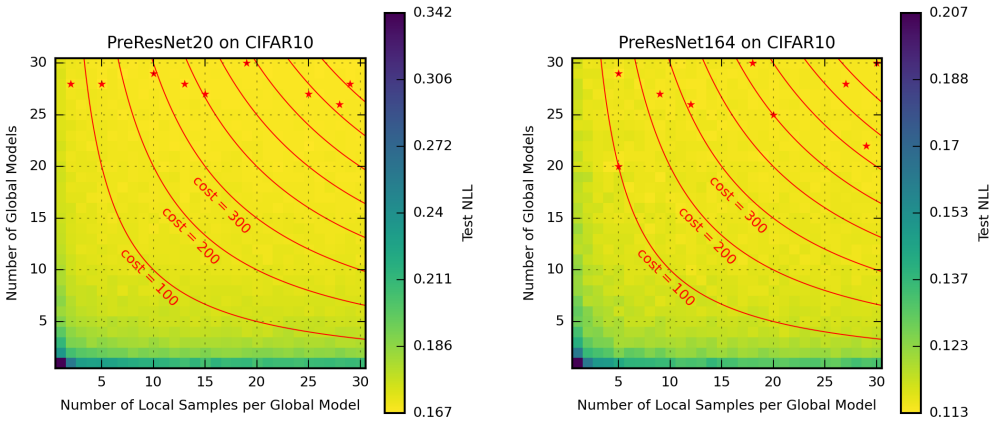


Figure 3.19: NLL Heatmaps from Figure 3.18, overlayed with lines of constant storage cost in terms of the number of stored parameter vectors for the ensemble represented by the given MultiSWAG solution. The red stars represent the optimal solutions within each band, i.e. with storage cost between the upper and lower line.

Figure 3.19 clearly demonstrates that the optimal solution within each band lies in the top few rows of the graph. This means that no matter how much storage space we have to spare, in order to minimise NLL, we should always prefer to dedicate it to representing global uncertainty via ensembling over local uncertainty via drawing local model samples.

MultiSWA Outperforms MultiSWAG

Our findings so far seem to suggest that representing global uncertainty corresponds to improving predictive performance, whereas representing local uncertainty leads to better calibration, and that these two dimensions are at odds with one another in MultiSWAG. Further, while we have found that MultiSWAG outperforms deep ensembles, we have found MultiSWA to substantially outperform MultiSWAG, in the sense that MultiSWA obtains a lower loss, higher accuracy, and is better calibrated. This is all despite being significantly less costly to store and to predict with. Figure 3.20 revisits the idea of diversity in order to offer a potential explanation, at least for the predictive performance angle of this result.

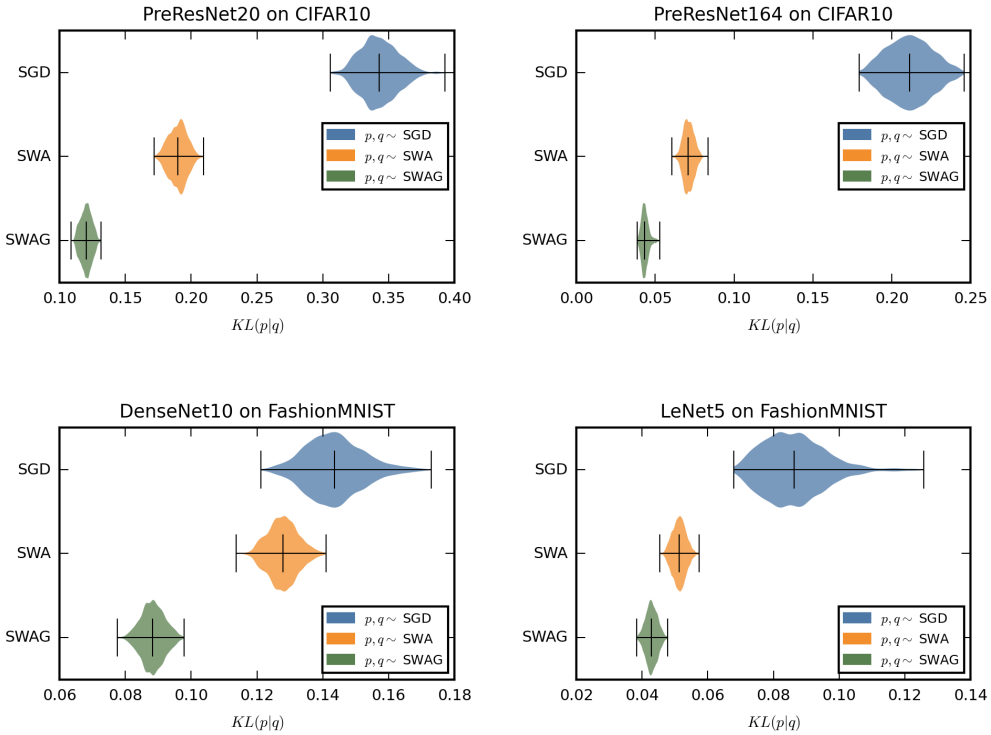


Figure 3.20: KL divergence densities observed between pairs of predictive distributions arrived at via SGD, SWA, and by approximately marginalising over a given SWAG solution (with 30 local samples).

In Figure 3.20 we plot the KL divergence density of the solutions arrived at by SGD, by SWA, and for each full SWAG approximation. To be clear: in contrast to Figure 3.12, this time we plot the KL divergence of the full predictive distribution represented by each SWAG model, i.e. via their full collection of local samples. We find that the diversity across complete SWAG models is substantially lower than that of SWA. Thus, despite their increased predictive performance, intuitively the SWAG models do not gain as much from ensembling as the SWA (or SGD) models.

Remarks

We have found that MultiSWAG is generally less performant than MultiSWA, and postulate that this likely has to do with the fact that each complete SWAG solution is significantly less diverse than each MultiSWA solution, at least in terms of predictive performance. The case for calibration is not immediately clear, but we offer an explanation for this in Section 4.1. While this result is somewhat disappointing for MultiSWAG, it does lead us to an interesting and important observation: despite the evidence that sampling models globally results in greater diversity than sampling models locally (e.g. Figure 3.12) , the diversity measured between distributions formed by local ensembles is surprisingly low.

Chapter 4

Conclusion

4.1 Discussion

We have conducted a thorough investigation of local and global parametric uncertainty through the lens of deep ensembles and SWA-Gaussian, in addition to their variants/hybrids. Through our experiments on in-distribution calibration and predictive performance, we have found that representing global uncertainty via ensembling substantially improves the predictions of a neural network, but overcorrects in terms of calibration, inducing underconfidence. In contrast, we found that representing local uncertainty (via SWAG) improves predictive performance to a lesser degree, but results in a better calibrated solution. We found that in combining local and global uncertainty in MultiSWAG, we obtained better performance than deep ensembles, but with similarly underconfident predictions. We found the MultiSWA solution, which represents global uncertainty of the SWA solution, to be more predictive, slightly less underconfident, and cheaper to store and compute than MultiSWAG.

Practical Recommendations

Based on our results, we would recommend to any practitioner who cared primarily about predictive performance to dedicate computational resources to a MultiSWA solution. Whereas, if a practitioner explicitly needs calibrated uncertainty estimates, we would recommend dedicating resources to a SWAG solution of sufficiently high rank/sufficiently many local model samples. These quantities may need to be determined empirically, but note that we do not suffer issues with overfitting from using a rank that is too high, since increased ranks simply allow for a fuller representation of the local posterior.

Underconfidence Matters

Throughout, we have been discussing underconfidence in almost the same vein as overconfidence. While the latter is certainly more directly dangerous for predictive models in control of automatic systems such as self-driving cars or medical equipment, the former comes with its own host of undesirable issues. For example, if a predictive model is consistently underconfident, then any system utilising its predictions will be limited to behave suboptimally. For example, if a medical diagnostics program is consistently underconfident and requires a high confidence threshold to make a diagnosis without consulting a human doctor, then the use of the system may be counterproductive. In a worst case scenario, a system could be implemented with the mentality that if *every* prediction is low-confidence, then *none* of them are, in which case we accumulate all of the problems with overconfidence. Indeed, the only robust solution is to create models that have well-calibrated predictive uncertainty.

The Role of Diversity for Predictive Performance

We found that the models sampled from within the local subspace described by SWAG were significantly less diverse than the models sampled globally from different modes. We refer to the theory presented in Tumer and Ghosh [56] for why ensembles benefit from diversity when predicting posterior probabilities from a pure ensembling perspective, though we note that this theory applies to the zero-one loss, and not the logarithmic loss. Thus, this provides an explanation for why MultiSWA and deep ensembles both obtain better *predictive* performance than the less diverse SWAG solution, given the same number of models. The role of diversity with respect to calibration is not explicitly known at this time [64].

Local Uncertainty is Better for Calibration than Global Uncertainty

We observed that a unimodal SWA-Gaussian solution was better at representing its own uncertainty than deep ensembles, MultiSWA, and MultiSWAG. While initially counterintuitive, there is a rather natural explanation for this. In constructing a SWAG model, we are essentially restricting our hypothesis space to be the space of predictive functionals represented by the mode that we are modelling via SWAG, i.e. assigning all solutions outside of this mode to have essentially 0 probability. Because the SWAG posterior is a reasonably good approximation to the true posterior within this subspace, the approximate Bayesian model average adequately represents our epistemic uncertainty in the possible predictive functionals *within* the mode. As soon as we include another mode, however, then we expand our hypotheis space, and the two isolated SWAG posteriors no longer represent good approximations to the true posterior over this hypothesis space, and in particular the Bayesian model average ignores epistemic uncertainty in the space between the solutions.

4.2 Future Work

In this work, we have investigated two particular flavours of local and global uncertainty. While we expect that our findings will generalise for global uncertainty, as ultimately any representation of global uncertainty will be some form of global ensemble, an interesting direction for future work would be to explore different notions of local uncertainty. While SWAG is simple and highly scalable, its representation of the local subspace is limited. Natural extensions to SWAG could be:

1. PCA SWAG: The default implementation of rank k SWAG holds onto the previous k deviation vectors, computing its subspace from these alone. We found that in practice the majority of these ranks were redundant, though this may have to do with the way in which the approximation is constructed. It would be interesting to see if the performance of SWAG could be increased by holding on to the deviations for longer, and computing the principle components of the subspace spanned by the SGD iterates at the end of training. This might lead to additional runtime cost, but would incur no overhead at test time. It may also be possible to implement this in an approximate manner such that it does not incur runtime cost.

2. Matrix SWAG: Intuitively, we might expect the weights in the same layer of a neural network to depend on one another more than weights corresponding to different layers. We may therefore be able to represent a richer and more useful covariance structure by assuming a block-diagonal form for the covariance matrix.

Furthermore, there are other practical representations of local uncertainty that would be interesting to investigate, e.g. [53, 31].

This work has also highlighted the need for a method which unifies global and local uncertainty in a way that maintains the benefits of both. Ideally we would like to be able to learn a calibrated model without sacrificing predictive performance, as ultimately this is the goal of predictive modelling. One potential direction of exploration is utilising the recent *Fast Geometric Ensembling* [20] as a method for walking around the high density regions of the posterior in a manner reminiscent of an MCMC sampler. Such an ensembler could traverse distinct modes and the low-loss tunnels that connect them continuously, collecting local models along the way, perhaps in a manner that could be tuned to be both predictive and calibrated.

References

[1] Tesla crash preliminary evaluation report, 2017. NHTSA. PE 16-007. Technical report, U.S. Department of Transportation, National Highway Traffic Safety Administration.

[2] Preliminary report – highway – HWY18MH010, 2018. National Transportation Safety Board, ("The information in this report is preliminary and will be supplemented or corrected during the course of the investigation").

[3] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*, 2020. arXiv: 2002.06470.

[4] K. Audhkhasi, A. Sethy, B. Ramabhadran, and S. S. Narayanan. Generalized ambiguity decomposition for understanding ensemble diversity, 2013. arXiv: 1312.7463.

[5] D. Barber and C. M. Bishop. Ensemble learning in bayesian neural networks. In *Neural Networks and Machine Learning*, 1998.

[6] J. M. Bernardo. Expected information as expected utility. In *The Annals of Statistics*, 1979. Theorem 2.

[7] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *In Proceedings of the International Conference on Machine Learning*, 2015. arXiv: 1505.05424.

[8] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars, 2016.

[9] G. Brown. *Diversity in Neural Network Ensembles*. PhD thesis, University of Birmingham, 2004.

[10] G. Brown and L. Kuncheva. "good" and "bad" diversity in majority vote ensembles, 2010.

[11] G. Brown, J. Wyatt, and P. Sun. Between two extremes: Examining decompositions of the ensemble objective function. In *Multiple Classifier Systems*, 2005.

[12] T. Chen, E. B. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. 2014. arXiv: 1402.4102.

[13] F. Draxler, K. Veschgini, M. Salmhofer, and F. A. Hamprecht. Essentially no barriers in neural network energy landscape. In *Proceedings of the International Conference on Machine Learning*, 2018. arXiv: 1803.00885.

[14] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In *Proceedings of the International Conference on Machine Learning*, 2019.

[15] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. 1987.

[16] A. Esteva1, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. In *Nature*, 2017.

[17] S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: A loss landscape perspective, 2019. arXiv: 1912.02757.

[18] M. Fortunato, C. Blundell, and O. Vinyals. Bayesian recurrent neural networks., 2017. arXiv:1704.02798. Presented at the 12th Women in Machine Learning Workshop.

[19] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[20] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, 2018. arXiv: 1802.10026.

[21] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. In *Neural Computation*, 1992.

[22] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. In *Journal of the American Statistical Association*, 2007.

[23] F. Gressmann, F. J. Király, B. Mateen, and H. Oberhauser. Probabilistic supervised learning, 2018. arXiv: 1801.00753.

[24] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the International Conference on Machine Learning*, 2017.

[25] L. K. Hansen and P. Salamon. Neural network ensembles. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.

[26] W. Hastings. Monte carlo sampling methods using markov chains and their application. In *Biometrika*, 1970.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016. arXiv: 1603.05027.

[28] G. E. Hinton and D. V. Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, 1993.

[29] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*, 2017. arXiv: 1704.00109.

[30] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2016. arXiv: 1608.06993.

[31] P. Izmailov, W. J. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, and A. G. Wilson. Subspace inference for bayesian deep learning, 2019. arXiv: 1907.07504.

[32] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In *Uncertainty in Artificial Intelligence*, 2018. arXiv: 1803.05407.

[33] G. James. Variance and bias for general loss functions. In *Machine Learning 51*, 2003.

[34] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, 2017.

[35] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proceedings of the International Conference on Machine Learning*, 2016. arXiv: 1609.04836.

[36] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *In Advances in Neural Information Processing Systems*, 2015.

[37] A. Krizhevsky, V. Nair, and G. Hinton. CIFAR-10 dataset, 2014.

[38] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017. arXiv: 1612.01474.

[39] Y. LeCun, B. Boser, J. Denker, R. E. H. D. Henderson, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, 1989.

[40] Y. LeCun, L. Bottou, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the Institute of Electrical and Electronics Engineers*, 1998.

[41] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra. Why m heads are better than one: Training a diverse ensemble of deep networks, 2015. arXiv: 1511.06314.

[42] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, 2018.

[43] W. Maddox, T. Garipov, P. Izmailov, D. Vetrov, and A. G. Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, 2019. arXiv: 1902.02476.

[44] S. Mandt, M. D. Hoffman, and D. M. Blei. Stochastic gradient descent as approximate bayesian inference. In *Journal of Machine Learning Research*, 2017. arXiv: 1704.04289.

[45] C. G. F. Naeini, Mahdi Pakdaman and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Association for the Advancement of Artificial Intelligence*, 2015.

[46] R. M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1996.

[47] R. M. Neal. Mcmc using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo, Chapter 5*, 2012.

[48] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the International Conference on Machine Learning*, 2005.

[49] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, 2019. arXiv: 1906.02530.

[50] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble method for neural networks. In *Neural Networks for Speech and Image processing*, 1993.

[51] B. Polyak and A. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 1992.

[52] R. Rahaman and A. H. Thiery. Uncertainty quantification and deep ensembles, 2020. arXiv: 2007.08792.

[53] H. Ritter, A. Botev, and D. Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

[54] K. Shridhar, F. Laumann, A. Llopart Maurin, M. Olsen, and M. Liwicki. Bayesian convolutional neural networks with variational inference., 2018. arXiv: 1806.05978.

[55] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. L. et al. Mastering the game of go with deep neural networks and tree search. In *Nature*, 2016.

[56] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. In *Pattern Recognition*, 1996.

[57] A. Tupasela and E. Di Nucci. Concordance as evidence in the watson for oncology decision-support system. In *AI Society*, 2020.

[58] J. Wang, Y. Ma, L. Zhang, R. Gao, and D. Wu. Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, 2018.

[59] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the International Conference on Machine Learning*, 2011.

[60] Y. Wen, G. Jerfel, R. Muller, M. W. Dusenberry, J. Snoek, B. Lakshminarayanan, and D. Tran. Improving calibration of batchensemble with data augmentations, 2020. Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

[61] A. G. Wilson and P. Izmailov. Bayesian deep learning and a probabilistic perspective of generalization, 2020. arXiv: 2002.08791.

[62] T. C. Yi-An Ma and E. B. Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, 2015.

[63] A. M. A. Zaher and A. M. Eldeib. Breast cancer classification using deep belief networks. In *Expert Systems with Applications vol. 46*, 2016.

[64] M. Zanda. *A Probabilistic Perspective On Ensemble Diversity*. PhD thesis, University of Manchester, 2010.