Radboud University Nijmegen

Faculty of Science

Uncertainty

Ensembles Of Deep Neural Networks As Predictive Distribution For Regression

Thesis MSC Artificial Intelligence

Author:
Thomas Manuel Rost
4469259

Supervisor:
dr. Johan Kwisthout

Second reader:
prof. dr. Maurits Kaptein

December 2019

Dedication

To my family, who somehow managed to understand the important parts, no matter what. Thank you.

Acknowledgements

I want to thank my Supervisors, Maurits Kaptein and Johan Kwisthout for their continuous help and support. This can't have been easy.
I'd like to thank Max Hinne for commenting on an early draft. The comments were appreciated.

Abstract

Recent years have seen the rise of Deep Neural Networks (DNN) and Bayesian Approaches in machine learning. Combining the mathematical expressiveness of DNNs with the quanti cation of their predictions' reliability through the Bayesian approach into *Bayesian Neural Networks* (BNN) promises a revolution for decision making both by humans and arti cial agents. However, certain theoretical and practical hurdles stand in the way of the reliable use of BNNs. This work aims to provide a primer on the theoretical problems encountered when building fully Bayesian Neural Networks and argues that the use of ensembles of DNNs can lead to a simple, practical substitute. To do so, we compare six di erent popular approaches to explicit and implicit ensembling of DNNs from the literature in the context of regression problems. We evaluate them on two synthetic and one real-life data sets with respect to the common metrics mean squared error (mse) and negative log predictive density (nlpd). Additionally, we introduce one metric that captures the correlation of the uncertainty of the predictive distribution on its error ('correlation between error and uncertainty,' cobeau). We focus on comparability between the methods by forcing them to ensemble a shared, independently determined network architecture with a predetermined training schedule in order to obtain their predictive distribution.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

'You're wrong{but even if you weren't wrong, you still can't do the computation.'

Frequentists to Bayesians, probably[1]

Recent years have seen the rise of Deep Neural Networks (DNN) and Bayesian Approaches in machine learning. Combining the mathematical expressiveness of DNNs with the quanti cation of their predictions' reliability through the Bayesian approach into *Bayesian Neural Networks* (BNN) promises a revolution for decision making both by humans and arti cial agents[1]. However, certain theoretical and practical hurdles stand in the way of the reliable use of BNNs. This work aims to provide a primer on the theoretical problems encountered when building fully Bayesian Neural Networks and argues that the use of ensembles of DNNs can lead to a simple, practical substitute. To do so, we compare six di erent popular approaches to explicit and implicit ensembling of DNNs from the literature in the context of regression problems. We evaluate them on two synthetic and one real-life data sets with respect to the standard metrics mean squared error (mse) and negative log predictive density (nlpd). Additionally, we introduce one metric that captures the predictive power of the uncertainty of the predictive distribution on its error ('correlation between error and uncertainty,' cobeau). We focus on comparability between the methods by forcing them to ensemble a shared, independently determined network architecture with a predetermined training schedule in order to obtain their predictive distribution.

Bayesian Deep Neural Networks

Bayesian Methods The history of the Bayesian approach to statistics is full of ups and downs. Invented sometime during the 1740s by a monk of the name Thomas Bayes and rediscovered by Pierre Simon Laplace, it is a framework that was initially designed to integrate new evidence with previously obtained experience to perform a coherent update of personal belief. The resulting subjectivity was, in the end, one of the major reasons why it faced decades-long opposition from theoreticians and policymakers alike[3].
In simple words, the central idea behind the Bayesian approach is described by `previous belief + new evidence = new and improved belief.' This simple concept can conveniently

[1] see, e.g., [2] for recent work on the topic of using BNNs to solve the problem of Exploration vs. Exploitation.

be formulated using Bayes' rule, leaving us with the probability distribution

$$P(\ jE) = \frac{P(Ej \ )P(\ )}{P(E)} \tag{1.0.1}$$

where is the *hypothesis*[2], E is our data or evidence[3], $P(\ )$ is the previous belief in the probability of the hypothesis, the *Prior*, $P(Ej \ )$ is the *likelihood* of the data given the hypothesis and $p(\ jE)$ is the *posterior*, the belief in the hypothesis after observing new evidence. The term $P(E)$ is called the *model evidence*, the prior probability of the data. It is conceptually slightly more taxing than the previous parts of the equation{as it is computed as the sum of the probability of the data given each possible hypothesis. As it turns out, the headache people get when rst trying to understand this concept has a tendency to stay for various reasons{and it plagued whole generations of Bayesians until the development of sophisticated sampling methods and the advent of powerful computing devices[4].

The big contender to Bayesian statistics are so-called `Frequentist' approaches, named after the belief that probabilities should capture naturally occurring frequencies rather than personal beliefs in the likelihood of an event occurring. While this claim to objectivity was one of the main reasons why Frequentist approaches had the upper hand over their Bayesian counterparts for a while, the additional computational complexity that comes with having to propagate full distributions (and thus often integrals) during inference in the Bayesian case compared to computing point estimators on a pre-collected sample in the Frequentist case played a signi cant role in giving proponents of the Bayesian theory a hard time.

However, with the advent of the information age and thus more capable computing machines as well as the discovery of Monte Carlo methods and variational approximation{as well as the release of previously classi ed material on code-breaking and other war-related mathematics{Bayesian Methods went through a resurgence. More and more practitioners and theoreticians recognized the bene ts of Bayesian statistics over Frequentist approaches for certain applications[5]. [4], for example, is generally considered one of the most in uential papers in sociology, and it introduces strictly Bayesian methods[5].

Neural Networks The discussions about which way of analyzing data is preferrable took place in a time when the interest in scienti c insight and data collection was at a height. Be it cosmology, epidemiology, nance, or neuroscience{every eld suddenly found a wealth of information that they needed to analyze with novel mathematical methods. This lead to many insights and new kinds of mathematical models in many elds.

In 1957, for example, [6] devised a new algorithm based on the mathematical description of neuroscienti c research of the time: the perceptron. This perceptron was made up of computational nodes that abstractly mimic the structure of the brain in so far as the strength of its output is derived from many input nodes[6]. It was, somewhat

[2] In our case the hypothesis often describes the likely distribution over parameters for a parameterized model

[3] Later comprised of a feature vector X and a target vector y.

[4] Which, as we will see, is a commonality shared with the other topic of this thesis, deep arti cial Neural Networks.

[5] The best stance to have in the ght between Frequentists and Bayesians seems to be one of the middles{both approaches tend to perform well on di erent problems. As we will see, this work mostly focussed on enabling Bayesian approaches employs some Frequentist methods such as Fisher's correlation coe cient.

[6] Which does not exactly mimic neural computations in the brain { spiking neural networks, which re only after a certain input threshold has been passed are a more realistic representation.

prematurely, declared to be the prototype of a machine that would soon begin to talk, walk, and learn like a human baby[7]. Soon after, however, [8] showed that a perceptron made up of a single layer could only learn linearly separable functions and thus were, for example, unable to learn the XOR function, and that was that for a long time.
Decades later, after research in Artificial Intelligence (AI) had been focussing on other directions[7], [9] showed that Neural Networks of a single layer using a sigmoid activation function could be used to approximate any continuous function[8]. [10] later proved this flexibility to be a property of the hidden layer structure rather than the choice of the activation function, laying the foundation for much of today's machine learning.
In a development that somewhat mirrors the rise of Bayesian methods, the potential of Neural Networks, too, was held back until the technology and mathematics to circumvent computational intractabilities{in the form of gradient descent{had caught up with its development. In this case, the breakthrough came when people realized that Graphical Processing Units (GPUs) were almost perfect for running the highly distributed mathematics that power the training of Neural Networks. Within a few years, Neural Networks rose to new glory. Latest when [11] showed the efficacy of Neural Networks on tasks such as image recognition they had proven their worth, rekindling interest in Neural Networks from research and industry[12].
Today, Deep Neural Networks (DNNs) are all around us. Whether the application is in parsing natural language ([13] and it's more efficient version introduced in [14], [15], [16]), deriving insights from data ([17], [18]), predicting medical conditions ([19]), producing and augmenting images, sounds and videos([20][9], [21] and [22] respectively) or enabling agents to act and play games ([23], [24] and [25]), they seem to power the future.

Bayesian Neural Networks These two approaches, both rising to the top of their respective fields in a similar time and connected through the quest for better predictive models, then surely have to be combined to form an even stronger alliance of informing scientists and leaders all over the globe{letting them make decisions based not only on a point estimate but also on the uncertainty that their model exhibits for its prediction. In the end, even the field that birthed DNNs is moving towards Bayesian treatments, both for data analysis as well as for a theoretical framework explaining the human mind[10].
Unfortunately, this is not currently the case, *Bayesian Neural Networks* (BNN), while getting more and more traction in research, remain a niche in the effort of finding new and better performing models; although they promise to solve problems beyond more accurate predictions[11]. As we shall see in more detail in later sections when we turn to Bayesian methods for predicting unseen values{which is generally seen as an area in which DNNs excel{we use the *Posterior Predictive Distribution*. This distribution specifies how new predictions $\hat{y}$ are distributed according to the distribution of the model's parameters.
Unfortunately, this distribution relies on two sources of computational complexity{the Neural Network as a model architecture and the posterior distribution over the parameters {and as we will see in section two of this work, this combination is hard to even approximate. Additionally, Bayesian methods are generally specified with regard to a *Prior distribution*{which, in the case of the ample, non-linear space of DNN parameters, poses a more conceptual challenge. This work will focus on a subspace of solutions to the first problem of intractability while leaving the problem of the prior specification to

[7] mostly logical programming languages and other logic-based approaches
[8] the so-called "universal approximation theorem"
[9] for examples see the fantastic `https://thispersondoesnotexist.com/`
[10] See e.g., [26]
[11] For an excellent introduction into how BNNs can increase safety in self-driving cars, for example, read the first few paragraphs of [27]

another time[12].

Related work

Previous work on obtaining a predictive distribution from Neural Networks is relatively vast; the following will provide an overview with typical approaches.

1. Predicting the scale parameter of the predictive distribution directly:
   In this approach, the neural network additionally outputs a node that represents the scale parameter of a distribution in addition to the location parameter. It is used in two distinctive ways: By turning Autoencoders into distributions over outcomes that can be drawn from, [29] and [30] and in a more traditional way for regression as, e.g., [31].
2. [32] Introduces dropout forward passes to turn previously deterministic Neural Networks into probabilistic models.
3. Ensembles themselves are often used as a method of regularization without the claim of approximating a posterior distribution simply because they work well in practice, see, e.g., [33] or [34]. They are also the basis of impactful approaches such as random forests, as discussed in [35].
4. Outside of Neural Networks, they have been used to produce posterior distributions with elaborate theoretical and practical motivation, e.g., [36].
5. Established methods such as Monte Carlo (MC) methods and variational inference (VI) methods have been tried on Neural Networks to derive a predictive posterior distribution, see, e.g., [37]. However, for reasons that go beyond the scope of this work, they either tend to underperform through a limitation of network parameters or become impractical for large networks and datasets relatively quickly.
6. An interesting approach to solving the previous problem comes from [38] (under review), who introduces a novel approach for stochastic variational approximation.
7. Finally, the use of a predictive distribution on practical applications is a broad field, a recent example is [39], in which the authors use a method called Gaussian Processes to obtain uncertainty over a space of possible experiments on fluid dynamics which in turn drives the setup for continuous experimentation.

Contribution

We aim to contribute to the field of approximate predictive distributions with DNNs in two ways. On the theoretical side, we provide intuition on how *ensembling* different DNNs can be seen as a valuable substitute for proper BNNs for problems of mean squares regression with a normally distributed target by directly assessing the predictive distribution and providing a description and overview of over six popular approaches of creating such ensembles. The practical contribution then is to compare these six approaches with regard to two classical measures of quality; the mean squared error

[12] However, interested readers are referred to, e.g., [28] for an in-depth explanation of the problem and some suggestions for solutions.

and the negative log predictive density as well as a novel[13] measure for the correlation between a model's uncertainty and its error. The comparison is done on synthetic- as well as real-life datasets and focussed on controlling as many variables as possible through the use of model architectures that are as similar to each other as possible.

Structure of this work

The remainder of this work is structured as follows: Section 2 aims to provide some theoretical background to the problem of obtaining predictive distributions from Deep Neural Networks, Section 3 introduces the ensembles used in this work.
Section 4 introduces the experimental setup, datasets, and models used, as well as the analysis, i.e., the measures and baselines used, as well as the motivation behind removing outliers from the experimentally obtained dataset. Section 5 contains empirical results derived from the experiments and a short analysis. A brief discussion of the results, as well as an outlook into future research, is given in Section 6. Section 7 recollects the most essential findings and conclusion. The appendix contains mathematical proofs and intuitions not fitting for the main body of this work as well as the exact specifications of hardware used and additional experimental results.

[13] While the authors could not find a definition in the literature, the absence of evidence is not to be taken as evidence of absence.

Chapter 2

Theoretical Considerations

This chapter will provide a general overview of the mathematics behind Deep Neural Networks, and intuition of why obtaining a predictive distribution from them is an active area of research. While some core concepts will be reviewed, the reader is assumed to be familiar with the basics of machine learning, statistics, and the Bayesian approach. The first part introduces the problem as well as a recursive description of DNNs. The second part introduces the posterior predictive distribution, from which we then intuit how the large parameter space in such models leads to computational and analytical intractabilities and explain the need for approximate methods to solve the problem of generating reliable posterior predictive distributions from DNNs. The last part explains some theory behind ensembling, and how to turn their predictions into a substitute predictive distribution as well as reasons for promoting diversity in ensembles.

2.1 Background

This section covers notation, problem statement, and a definition of Deep Neural Networks and their training objective.

Notation

We introduce the notation used in this work. We assume familiarity with mathematical conventions when it comes to notation and will only introduce cases with most frequent use or where our notation carries additional information not covered by the convention.

Basics Scalars are non-bold, lower-case letters such as, e.g., y unless they indicate dimensionality in which case they are upper case, e.g., N. Upper case letters are overloaded to indicate sets depending on the context such as $D = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$. We deviate from this notation in the case of θ, a lower case, greek letter that is defined to indicate the set of parameters of a neural network. Vectors are lower case, bold letters such as $\mathbf{x}$, matrices are upper case, bold letters such as $\mathbf{W}$.
$\hat{y}$ indicates a prediction for y. Data are represented as row vectors and drawn from the underlying population, e.g. $\mathbf{x} \in \mathbf{X}$, independently and identically distributed.

Distributions $p(x)$ indicates a probability density distribution over x, $p(y|x)$ indicates that the distribution over y is conditional on the observation of x. In case x is emphasized to be a point estimator rather than a distribution, we indicate that by writing $p(y; x)$. E[x] indicates the expected value, or expectation of x, which is a (possibly

weighted) average over the entries $x_i \in \mathbf{x}$. $\mathrm{E}[y|\mathbf{x}]$ indicates the expected value of y given the input $\mathbf{x}$, $\mathrm{E}[\mathbf{x}]$ indicates the expected value is computed given some parameters of the underlying model, .

Models, Ensembles, and Members Given the ambiguity of the word `model' in a hierarchical context{it would be fair to refer to an ensemble as a `model,' but in the same way, the members that make up its prediction are `models'{we decided to di erentiate between three, more speci c words:

1. `Ensemble' will refer to a collection of M (potential) models that are trained to solve the same problem and then combined in some way to obtain the solution. Mathematically, it is described as a set of M sets of parameters $_m$, $\{ g_m^M$.
2. `Member' or `Ensemble Member' either refers to an instantiation of a model comprising an ensemble or a draw from a distribution over the parameter space spanned by the ensemble. In any way, in our work, it is a set of parameters $_m \in \{ g_m^M$
3. Further, the `model' will refer to any mathematical model that is being used to model some output. We will use it in contexts where the di erentiation between Ensemble and Member either is given by context or irrelevant (or in cases where we refer to neither Ensemble nor Members).

Problem statement

We assume a training data set D comprised of N independent and identically distributed data points $D = \{\mathbf{x}_n; y_n\}_{n=1}^N$ drawn from a true data distribution $p(D| _{origin})$ dependent on unknown parameters $_{origin}$ where $\mathbf{x} \in R^F$ are the F dimensional features, and y is assumed to be real, $y \in R$. We further assume the target y to be normally distributed conditional on the feature vector x and the parameters $_{origin}$, $y \sim N(_{theta_{origin}}(x); _{origin})$ with unknown noise $_{origin}$.
Given the input vector $\mathbf{x}$, we use ensembles of M DNNs comprised of L layers to predict a distribution over the target $\hat{y}$, $p_{\{ _m\}_{m=1}^M}(\hat{y}|x)$. The parameters $\{ _m\}_{m=1}^M$ where $= \{\{\mathbf{W}_l\}_{l=1}^L; \{\mathbf{b}_l\}_{l=1}^L\}$ with $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weights and biases of layer l, of the ensemble members $_m$ are obtained in various di erent ways as described in 3.5.

Recursive Description of Deep Neural Network's Expectation

Equations (2.1.1) show the recursive de nition of the expectation E of a Deep Neural Network of L layers towards the predicted target $\hat{y}_i$ given given the input $\mathbf{x}_i$ and the network's parameters as a set of weights and biases $\{\mathbf{W}_l\}_{l=1}^L$ and $\{\mathbf{b}_l\}_{l=1}^L$ for each layer $l \in L$. The mathematically more rigorous derivation of this equation from generalized linear models following [28] is presented in Appendix B.
This de nition will help us in determining what options we have for generating the predictive posterior.
We de ne the output of our neural network of L layers as the expected value of a distribution over predicted target values $\hat{y}$:

$$\mathrm{E}_{\{\mathbf{W}_l\}_{l=1}^L;\{\mathbf{b}_l\}_{l=1}^L}[\hat{y}_i|\mathbf{x}_i] = g^{-1}(h_L(\mathbf{x}_i; \{\mathbf{W}_l\}_{l=1}^L; \{\mathbf{b}_l\}_{l=1}^L)\mathbf{W}_{L+1} + \mathbf{b}_{L+1}) \tag{2.1.1}$$

where the hypothesis h_l is recursively de ned for each $l \in [0; L]$ as

$$h_l(\mathbf{x}_i; \{\mathbf{W}_j\}_{j=1}^l; \{\mathbf{b}_j\}_{j=1}^l) = f_l(h_{l-1}(\mathbf{x}_i; \{\mathbf{W}_j\}_{j=1}^{l-1}; \{\mathbf{b}_j\}_{j=1}^{l-1})\mathbf{W}_l + \mathbf{b}_l) \tag{2.1.2}$$

The end of the recursion happens when we ran through all the layers; in the end, we plug in the training sample itself:

$$h_0(\mathbf{x}_i) = \mathbf{x}_i \tag{2.1.3}$$

Going forward, the model's weights and biases are merged into the set describing all the parameters of our neural network, $= f f \mathbf{W} g_{l=1}^{L}; f \mathbf{b} g_{l=1}^{L} g$, to free the notation from unnecessary clutter. Note that in our case of regression the output link, g^{1}, is simply the identity function (and for the gradient descent its inverse{the identity function!). f_l are generally non-linearities. Popular choices include the Recti ed Linear Unit (ReLU), leaky ReLu, Tangens Hyperbolicus as well as the sigmoid function. All these functions have di erent bene ts and pitfalls, for an in-depth review see, e.g., [40].

Training objective

To train this Neural Network, we de ne a training objective that is used to nd a set of parameters that optimizes this objective via some form of gradient descent.
We use the mean squared error (mse):

$$= \operatorname{argmin} \frac{1}{N} \sum_{i=1}^{N} (y_i \quad \mathrm{E}[\hat{y} j x_i])^2 \tag{2.1.4}$$

For least squares regression it can be shown that minimizing the mse yields the same parameters as minimizing the Kullback-Leibler Divergence D_{KL} between the the empirical distribution over the data depending on the hidden generating parameters $_{origin}$, $p(D j _{origin})$, and the distribution over the data given the model parameters $p(D j)$ [1][41],[42].

$$= \operatorname{argmin} D_{KL}(p(D j _{origin}) j j q(D j)) \tag{2.1.5}$$

where D_{KL} is de ned as

$$D_{KL}(P j j Q)) = \sum_{i}^{N} p(x_i) \log \frac{p(x_i)}{Q(x_i)} \tag{2.1.6}$$

We are thus free to de ne with regards to this measure depending on the original choice of and $p(D j _{origin})$ which will be useful in de ning the implementations of our ensembles.

2.2 Posterior predictives and analytical solutions

From the posterior distribution over the parameters , as de ned in equation (1.0.1), we look towards a predictive posterior distribution to turn our beliefs about the hypothesis into predictions of our target variable y:

$$p(\hat{y} j x; D) = \int p(\hat{y} j x;) p(j D) d \tag{2.2.1}$$

where $p(\hat{y} j x;)$ is the distribution over the estimated target $\hat{y}$ evaluated at the point x given the parameters of a model and $p(j D)$ is the posterior distribution over these parameters given the training Data. We marginalize over the parameter space to obtain

[1] see B for an intuition

$p(\hat{y}|x; D)$, the distribution over the estimated target evaluated at a novel data point x given the training Data D.
While it is easy enough to write this equation down, two problems are apparent: We do not know how to specify a prior over an ensemble, and the dependency on the posterior $p(\ |D)$ is problematic. Evaluating this distribution becomes intractable for large parameter spaces[2]. In general, we have three options to evaluate this equation to obtain a valid distribution over plausible output values:

Analytical solution

For a limited set of problems, the posterior distribution can be computed analytically via a closed-form expression. This set of problems is de ned in a way that prior and posterior distributions are compatible with each other, more speci cally by being part of the same family of probability distributions. An example of an analytically solvable problem through so-called *conjugate* priors are the beta-binomial distribution and the Poisson-Gamma.
Unfortunately, only certain distributions of particularly simple makeup have known conjugates. Deep Neural Networks generally have no known conjugate priors. Specifying a prior over DNNs is a complex matter in itself. Were they known, closed-form analytical solutions for GLMs with a link function other than the identity are generally not known even for the case of point estimates[3]. Since our description in (2.1.1) shows our neural network to be a stack of GLMs where only the output layer employs the identity function, nding such a solution seems unlikely.

Variational solutions

Variational inference is a way of providing an analytical expression to an approximation of the posterior distribution through optimization[44]. The general idea is that a known family of parametrized distributions Q, such as a Gaussian, can approximate the distribution underlying the `true' and unknown distribution p. An `optimal' instance of this family, $q(\ )$, can then hopefully be found by minimizing a measure of dissimilarity D between p and Q. In the case of our predictive distribution, this can be expressed as follows:

$$p(\hat{y}|x; D) = \int p(\hat{y}|x; \ )q(\ )d \tag{2.2.2}$$

$$q(\ ) = \underset{q(\ )2Q}{\operatorname{argmin}} D(q(\ )||p(\ |D)) \tag{2.2.3}$$

The most common choice for the dissimilarity D is the Kullback-Leibler Divergence D_{KL}, as de ned in equation (2.1.6)[4].
For DNNs, this is usually done by assuming the distribution over weights to be normal. These normal distributions can then be tackled either analytically, e.g., by factorization such as performed by [45] or numerically, such as e.g., presented in [46].
The downside of the variational approach{apart from the possibility that q fails to capture important properties of p, e.g., when approximating a multi-modal distribution with a normal{is that its derivation is quite nicky in all but the most basic cases. This often leads to replacing one intractable distribution with a slightly less so, but still intractable approximation. [45], for example, is only computable in cases of very

[2] For an intuition of where the complexity arises, see Appendix B
[3] However, for certain classes of GLMs, there seems to be, such as provided by [43]{so ngers crossed
[4] The KL divergence is not symmetrical. In practice, $D_{KL}(q||p)$ has more benign properties and is used more often.

small Neural Networks; however, it is still the basis for many more recent approaches to variational inference in BNNs[47]. At the moment of writing, we are not aware of a general, explicit and e cient description of variational inference for the case of Neural Networks, although works like [48] provide valuable insights into the topic.

Sampling solutions

An alternative to the analytical description of the posterior predictive distribution or its variational approximation is by generating multiple plausible draws from the posterior distribution to approximate its shape[49].

$$p(\hat{y}|x; D) = \int p(\hat{y}|x;)p(|D)d \tag{2.2.4}$$

$$p(\hat{y}|x; ^{(s)}); \quad ^{(s)} \quad p(|D) \tag{2.2.5}$$

This then allows us to approximate metrics of our distribution easily, as for example the expectancy: $E[p(\hat{y}|x; D)] \quad \frac{1}{S} \sum_{s=1}^{S} p(\hat{y}|x; ^{(s)})$. Generally, to do this, we need a probabilistic model that can provide us with such draws, by repeatedly querying the output for a speci c input value. These draws are assumed to approximate the underlying distribution with enough samples. This process is referred to as Monte Carlo methodology and is a widely used process in statistics.
As we can see in (2.1.1), this description of a DNN will not be able to provide us with such draws: it is deterministic, and thus, we will not be able to sample a distribution over values by repeatedly querying it for a speci c x.
However, a technique that has long been used to derive more exact predictions via a simple combination of models uses an approach that is related in that it approximates a distribution over predictions from many point estimators. The following section will explain the basics of ensembling in Neural Networks.

2.3 Ensembles as a predictive distribution

Definition

The de nition of an ensemble varies more or less slightly depending on the context in which it is discussed[5]. In this work, we refer to an ensemble as a collection of M models with parameters $\{ _m\}_{m=1}^{M}$ that can generate predictions on a data set of interest. These *ensemble members* are being considered when making the nal prediction of the ensemble itself. In the case of a regression problem, this is usually done by averaging over their predicted values in order to obtain a predictive mean that tends to perform better than any member itself[6], for example as in (2.3.2). Indeed, ensembling Neural Networks seems to work particularly well, so much so, that in many competitions such as on kaggle or the imagenet challenge, ensembles of Neural Networks are usually among the winners.
When using ensembles to derive a predictive distribution, we can simply compute the metrics of the distribution we are looking for, in our case the Normal which we then

[5] It was used by [50] to describe a large (or in nite) number of states describing a possible system{in essence, a probability distribution over the state of the system

[6] for an explanation of voting in ensembles for classi cation and how it lowers the likelihood of assigning a wrong class, see, e.g., [51]

treat as our predictive distribution:

$$q(\hat{y}_i|x_i, D) \quad \mathcal{N}(\mu_{\{f_m\}_M^M}(x_i), \sigma_{\{f_m\}_M^M}(x_i)) \tag{2.3.1}$$

$$\mu_{\{f_m\}_M^M}(x_i) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\theta_m^*}[y_i|x_i] \tag{2.3.2}$$

$$\sigma_{\{f_m\}_M^M}(x_i) = \sqrt{\frac{\sum_{m=1}^{M}(\mathbb{E}_{\theta_m^*}[y_i|x_i] - \mu_{\{f_m\}_M^M}(x_i))^2}{M}} \tag{2.3.3}$$

where $q(\hat{y}_i|x_i, D)$ is the predictive distribution over the ith data point $\hat{y}_i$ with location parameter $\mu_{\{f_m\}_M^M}(x_i)$ and scale parameter $\sigma_{\{f_m\}_M^M}(x_i)$. $\mathbb{E}_{\theta_m^*}[y_i|x_i]$ is the expectation of the m-th DNN in our ensemble as de ned in (2.1.1) with it's set of parameters θ_m independently obtained according to our de nition of mse in (2.1.4):

$$\theta_m = \text{argmin}\frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbb{E}_{\theta_m}[\hat{y}|x_i])^2 \tag{2.3.4}$$

which we can express in terms of the KL divergence as we did in equation (3.4.2):

$$\theta_m = \text{argmin}\, D_{KL}[p(D|_{origin})||q(D|\theta_m)] \tag{2.3.5}$$

While this work will not explore the theoretical foundations of how to specify Priors for ensembles of DNNs upfront, it should be noted that for ensembles derived via this method, an *empirical Prior* exists, which we can compute by simply computing the mean and the standard deviation of the distribution over the members *before training*. We will use this in the later sections of this work.

Diversity

The e ectiveness of ensembles of DNNs in practice compared to approaches utilizing a single model seem to stem from the di erent strengths and weaknesses of their individual members.

An intuitive explanation for this behavior can be found when considering that the number of possible local minima grows exponentially with the parameter space[52] and that local minima often have comparable error rates when averaged over the whole data set while di ering in *where the error occurs*. DNNs usually have vast parameter spaces, and thus it is likely that di erent models would tend to converge towards di erent such minima of comparable average errors while making di erent individual mistakes[53]. As we can see in our distribution and especially in equation (2.3.2), these individual errors{ assuming they are distributed around the true target in an unbiased manner{would average out and give the ensembles an advantage over each singular model.

As we can see, the quality of this approximate posterior depends largely on both the t of the parameters θ_m to the data as well as their *diversity*. Their dependency on su ciently di erent members is a well-known property of any ensemble. Additionally, the tendency to agree or disagree of di erent but equivalent models as computed in (2.3.3) can intuitively be interpreted as the uncertainty of the ensemble.

The next section will introduce ways of encouraging diversity among ensembles of neural networks.

Intuition

While Bayesian prediction itself can be seen as a form of averaging over an in nite model space where the ensembles themselves are weighted by the likelihood of their parameters

(see, e.g., [54] for further reading), the link between deriving a predictive distribution from several independently trained Neural Networks and proper Bayesian inference is not conclusively established (see e.g., [47] for an in-depth review and [55] for a current attempt to solve the problem).
There seem to be two major problems with deriving Bayesian networks from Ensembles: The speci cation of a *prior distribution* over the large and non-linear parameter space of the ensemble itself is not a trivial task, although the literature on this topic is continually growing. For an in-depth review, see, e.g., [28].
The other problem is that inference over the *posterior distribution* is impossible for ensembles, again through the non-linear combination that happens in DNNs{it is not possible to average over the parameters as we have done for the output.
However, through the use of KL divergence to optimize the parameters of the ensemble members might point towards a link between the distribution spanned by the ensemble and variational inference towards the true data distribution $p(Dj_{origin})$. While the scope of this work limits the attention this link receives here, it is an intriguing direction for further investigation.

Chapter 3

Ensembles

This section gives a short reasoning for the choice of ensembles used in the experimental section of this work as well as an introduction of the individual methods. We will introduce their origin, including the original publication and possible amendments made to facilitate comparability in the experimental setup, the mathematical description of the origin of their diversity by utilizing equation (3.4.2), as well as notes on their practical implementation.

3.1 Comparability of architectures

Ensembles are a complex and powerful tool for improving the predictive quality on a data set, and as we shall see, to obtain a predictive distribution over the target. In theory, there are few restrictions on what kinds of architectures can be used as members of an ensemble. Even di erent classes of algorithms can be sampled together, using their respective strengths and weaknesses to build ever stronger models. This work, however, will focus on ensembles made up entirely of DNNs, and all members within an ensemble were treated the same way during training.
Unfortunately, even ensembles made up of one class of members are very hard to compare in a generalizable way. While in practical situations where a problem needs to be solved, a grid search can yield a set of parameters that is `optimal enough,' in a comparison like ours, it is not enough to pitch arbitrary subsets of parameters and training regimes against each other.
The reason for this is: Di erent methods might react di erently to underlying architectures. An approach, as proposed by [31], for example, is likely to perform well within the parameters that are usually used for a certain problem, given that it only introduces small and passive changes to classical DNN theory. An approach like [46] on the other hand, which uses a custom loss function as well as additional parameters to capture the variation of its weights is by no means guaranteed to function optimally when treated as a classical DNN and might well show its best outcomes in rarely encountered regions of the architectural space.
Optimizing each approach separately, which would work su ciently well for practical applications, also is not an option that satis es comparability. A reason for this is that it is not clear on which measure this optimization should be carried out. A low score for mean squared error on a model does not give us information about the performance of its uncertainty at all, and as we shall see in the experimentation 5, di erent metrics capturing di erent qualities of the predictive distribution can vastly disagree on the t of the distribution. Thus, an architecture and training regimen leading to similar outcomes for one metric still cannot be said to be unanimously comparable.

This problem shines through the lines of many publications; it is, however, made explicit in at least one: While [56] reports their method breaking several benchmarks and even surpassing explicit Variational Methods, [48] criticizes the comparison as potentially unfair, mentioning that the VI approach was not used in an optimal fashion which might have given it the edge over the ensemble.
Our solution to this conundrum is to only include ensembles that can reasonably be assumed to be `similar enough' to make a comparison fair. The criterion we used to select ensembles for our comparison is that they need to be expressible through a change to their training criterion, as de ned in (3.4.2). An extension of this criterion can be expressed as `each ensemble in the comparison has to be de ned through changes to either the distribution $p(Dj_{origin})$ or $q(Dj_{m})$.' Since the choice of the class of distribution for q is the normal, the ensembles have to be expressed by augmenting either the training data set D, their choice of initial parameters $_{m}$ or by post-processing the optimal value for it, $_{m}$.
While this is by no means a perfect approach, it at least ensures that the results obtained through our experiments are obtained on a shared architecture, which is likely in a similar level of optimality for each ensemble.

3.2 The base model

All ensembles are based on the same DNN architectures, with one speci cation for each dataset in the comparison de ned by the set of parameters $f\ g$ and trainable via minimizing the mse. This approach limits the selection of ensembles in such a way that they can only be included if their general idea can be expressed in terms of these parameters. We feel it actively aids comparability between ensembles as comparing di erent architectures of Neural Networks is a non-trivial task given their complexity[1]. The parameters of this base model were determined by a grid search limited to reasonable values for layer sizes in regression problems, non-linearities, and other parameters, such as the decay rate of the optimizer. It is implemented as an extension of Pytorch's nn.module[57], trained via Adam optimizer[58] and initialized via Pytorch's default initialization scheme[59] The number of epochs was determined empirically by closely monitoring the test set loss and nding convergence, then rounding to the nearest clean integer.

3.3 Multi-model ensembles

This class of ensemble is characterized by training M di erent instances of a base model, which are then combined in a straight forward fashion as described in (2.3.1). The number of models M is 10, following literature on similar topics. They are described in [60] or [61] and have long been a staple of ensembling techniques.

[1] Imagine two vastly di erent DNNs{we would like to compare them at similar levels of e ciency, which is determined by hyperparameter tuning. Even with the best methods, we could not be sure that we nd a comparable set of hyperparameters for each architecture and would have to rely on empirical measures. However, optimizing an architecture on one measurement does not mean that it is also co-optimized for another measure. For some models, such as the snapshot model, we can, to some extent, circumvent this problem by taking the idea originally expressed in their inception and changing it up slightly to arrive at a model that works on comparable intuition. Models like the one introduced by [46], while conceptually interesting and practically proven, have to be excluded from this comparison for reasons of architectural incompatibility.

Mult-initialisation ensemble

This very basic ensemble derives its diversity simply by using M di erent initialization values for $_m$. Note that all the other multi-model ensembles inherit this behavior without explicitly being stated.

$$_m = \text{argmin}\, D_{KL}[p(Dj\ _{origin})jjq(Dj\ _m)]; \qquad _m \quad U(\ ^{p}_{k};\ ^{p}_{k}) \tag{3.3.1}$$

where U denotes the Uniform distribution and $k = \frac{1}{F}$ where F is the number of features[59].

Fixed data shuffling

This ensemble is derived from the initialization ensemble by adding and locking random shu ing of the training data for each ensemble member.

$$_m = \text{argmin}\, D_{KL}[p(D_m j\ _{origin})jjq(D_m j\ _m)] \tag{3.3.2}$$

where D_m is a random shu ing of the original dataset D xed for each set of parameters $_m$.

Bootstrap sampling

The bootstrap is a very established technique in statistical modeling. The idea is to generate M di erent training sets $fD_m g_m^M$ generated by sampling from the original training set D with replacement.
In our implementation, due to the resampling, the class initialization additionally requires the length of the data set.

$$_m = \text{argmin}\, D_{KL}[p(D_m j\ _{origin})jjq(D_m j\ _m)] \tag{3.3.3}$$

where D_m is a randomly drawn resampling of the original dataset D xed for each set of parameters $_m$.
Interestingly, [62] highlights similarities between bootstrapped ensembles and the posterior distribution of Bayesian methods, even more speci cally, [63], chapter 8, p 261 mentions that samples obtained via a bootstrap ensemble are `a poor man's' posterior{ since it approximates the distribution well enough. Noteworthy is that [64] devised a method for bootstrapping in real-time. While Neural Networks generally do not perform very well with on-line methods, as soon as we nd out how to handle this problem, we will be able to derive expressive models based on the bootstrap on the y.

3.4 Snapshot based ensembles

The snapshot ensembles are two variations on a technique devised by [56]. In the original speci cation, the authors trained a neural network with a learning-rate decay cycle that was reset whenever the model converged. Whenever this happened, the authors copied the current state of the network, reset the learning rate to a high value, and started the training again. They report training gains through hopping from one local minimum to the next in addition to `train[ing] 1, get[ting] M for free'{a reference to the fact that the snapshots generated in this approach were used to form an ensemble that performed well on their benchmarks while saving computational cost by only training one network instead of M. The particular training schedule introduced in the original paper would

pose a problem to our goal of model comparability because most of our models are only trained to converge once. To overcome this limitation, we implemented two slightly di erent schemes utilizing the snapshot nature of the underlying publication.
Note that for this type of ensemble, $_m$ is replaced with $_t$ in the mathematical descriptions to emphasize the time that passes between epochs. We take a snapshot every $\frac{epochs}{20}$ epochs while using only the last ten saved models to generate the ensemble, resulting in a comparable set of predictions to the other ensembles introduced in this work.

Snapshot ensemble

This ensemble uses the scheme de ned above, taking snapshots every few epochs and saving them to disk. During experimentation, the last ten snapshots are loaded and initialized, and their predictive distribution computed as usual.

$$_t = \text{argmin} D_{KL}[p(D|_{origin})||(q(D|_{t\ 1})]; \qquad _0 \quad U(\ \bar{p}_k; \bar{p}_k) \tag{3.4.1}$$

where $_{t\ 1}$ is the previous snapshot, $_0$ is the randomly initialised set of parameters and $_t$ is the most recent set of parameters found by going through the full cycle of epochs.

Bobstrap

The Bobstrap was devised after a novel ([65]), which explores a set of main characters who are repeatedly digitally cloned at several points in time. Each clone's experience di ers from the others with more or less slight variations, which leads to them coming to di erent conclusions and developing sightly di erent characters. In our implementation of this idea, each time a snapshot is taken in the same way as in the snapshot ensemble, the data is additionally resampled with replacement as we would do for the bootstrap.

$$_t = \text{argmin} D_{KL}[p(D_t|_{origin})||(q(D_t|_{t\ 1})]; \qquad _0 \quad U(\ \bar{p}_k; \bar{p}_k) \tag{3.4.2}$$

where D_t is a full sample of the training dataset D with replacement, $_{t\ 1}$ is the previous snapshot, $_0$ is the randomly initialised set of parameters and $_t$ is the most recent set of parameters found by going through the full cycle of epochs.

3.5 Dropout

Dropout is a technique originally devised for the regularization of neural networks. In essence, it randomly excludes nodes from the network (*dropped out nodes*) during training, thus avoiding over tting and reliance on single nodes in the network [66].
[32] used this established technique to derive a posterior predictive distribution from a Neural Network by keeping the dropout for the prediction pass as well, rendering the DNN probabilistic depending on the dropout[2].

$$_{dropout;m} = \ _m \ \ Filter \tag{3.5.1}$$

$$_m = \text{argmin} D_{KL}[p(D_m|_{origin})||q(D_m|_m \ \ Filter)] \tag{3.5.2}$$

where $Filter$ is a matrix of the same dimensionality as $_m$ where the corresponding entries to the weights of the Neural Network are replaced by draws from the Bernoulli distribution with probability $p = 0.05$, resampled each time a prediction is made.

[2] While there is a discussion ongoing on whether the probabilistic nature of the dropout leads to proper posteriors (see, e.g., [67]). As some other discussion about risk vs. uncertainty (see, e.g., [68]), for the very modest goal of this work, these are not relevant.

Chapter 4

Methods

This section describes the experimental setup, such as datasets and data preparation, the parameter selections for the base model used for the ensembles, the use of baselines, and measures taken to ensure reproducibility. It also explains the measures used to quantify the di erence in predictive uncertainty as well as the decision process for the removal of outliers. The complete list of software and hardware used can be found in appendix A. The reader is assumed to be familiar with the basics of machine learning experimentation.

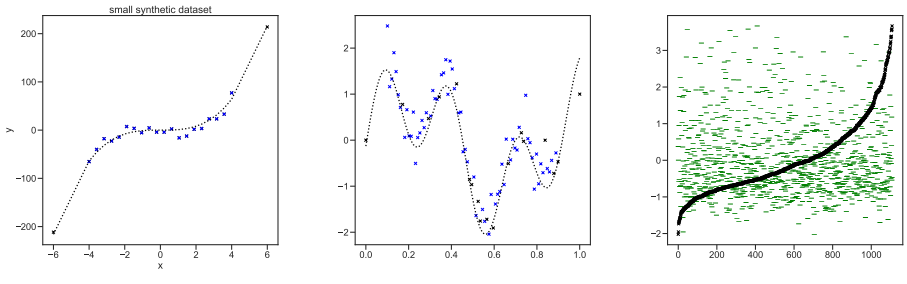
4.1 Experimental Setup

4.1.1 Datasets

We use two synthetic datasets, one for visual inspection inspired by [69], generated by the function $f_{small}(x) = x^3 + _{small}$ where $_{small}$ $N(0; 9)$ containing 20 data points in the interval [-4,4]. This dataset is used to inspect the out of sample distribution behavior of the models on a simple dataset. The test set is thus equal to the training set with added values at -6 and 6. A visualisation can be found in gure 4.1a
A second synthetic dataset was generated by the function $f_{large}(x) = sin(x \quad 20) + sin(x \quad 7{:}5) + _{large}$ where $_{large}$ $N(0; 0{:}3)$ for the interval from [0.1,0.9]. The experiments are conducted with and without out of sample data points at 0.0 and 1.0 in the test set to add out of sample data. This dataset with one predictor variable was mainly used in order to test the performance on a dataset with known parameters. A visualization can be found in 4.1b.
Additionally, we used one real-world regression dataset based on [70] to test our models in a more complex, multidimensional case. Since the goal of this work is not to challenge state-of-the-art predictive power on this dataset, we removed all non-numeric feature columns from the dataset to avoid elaborate pre-processing, leaving us with 37 predictors- and one target variable. For visualizations, the dataset has been sorted by target value to aid understanding, see 4.1c.
Both datasets have been scaled by subtracting the mean and standard deviation to obtain a more Neural Network friendly dataset (as suggested by, e.g., [71]).

4.1.2 Reproducibility

We performed 20 di erent train/test splits and shu ing at random with a training data size of 80% of the dataset and a test size of the remaining 20%. Given the goal of this work and the fact that at no point were the models' parameters repeatedly optimized



(a) small synthetic dataset (b) large synthetic dataset (c) real life housing dataset

Figure 4.1: *Data sets used in the empirical evaluation. Fig 4.1a shows the small synthetic dataset generated by the function* $y = x^3 + N(0; 9)$*, training data evaluated on [-4,4], test data interval [-6,6]. Dotted line is the ground truth. 4.1b shows the large synthetic dataset constructed via* $y = sin(x \quad 20) + sin(x \quad 7{:}5) + N(0; 0{:}3)$ *and evaluated on [0,1] Dotted line is the ground truth. 4.1c shows the real life housing dataset; y is the target house price mean centered and standardised by dividing through the data standard deviation. X is an indicator rather than the 37 dimensional feature space. Green indicates original sorting, black is sorted by ascending y value for better visualisation.*

on the measures of interest, we decided to forego a hold-out validation dataset.
Each experimental outcome was xed via a di erent random seed to ensure reproducibility. Note that these seeds mainly in uence the probabilistic parts of the model, i.e., the initialization matrices of the Neural Network as well as the bootstrap samples, the data shu ing and split, and the dropout where applicable.

Base Model Parameters The base model used for the ensembles is a feedforward DNN. Table 4.1 shows the nal choice of architecture for each data set. The parameters were found by grid search over a space restricted by reasonable choices for regression models from the literature. Dropout was only applied to the dropout ensemble, where it was added before each nonlinearity in accordance to the literature [47].

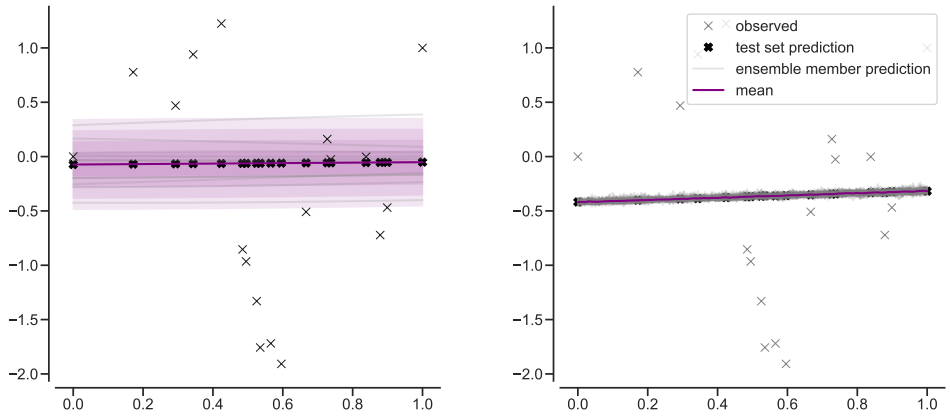
	small synthetic	large synthetic	housing
layer sizes	[500,300,200,10,1]	[100, 100, 10, 1]	[500, 500, 15, 1]
non-linearity	ReLU	Tanh	Tanh
optimizer	adam, decay 0.001	adam, decay 0.0005	adam, decay 0.005
epochs	100	500	300
Bootstrap Probability	0.7	0.7	0.7
Dropout Probability	0.05	0.05	0.05

Table 4.1: Network architectures for synthetic and real world datasets

4.1.3 Baselines

We compare the models' outputs to one data derived baseline that uses the empirical mean standard deviation of the training target, as well as the models' empirical prior

distributions where available[1]. A visualisation of the two priors used can be found in gure 4.2. In the case of the large synthetic dataset, we included the data generating process de ned by the data generating function $f_{large}(x)$ fo compare mse and nlpd to this `gold standard.' Note that because for this function the uncertainty is constant, cobeau can not be computed.



(a) ensemble prior (b) dropout prior

Figure 4.2: Typical priors for di erent ensemble classes on the large synthetic data set

4.2 Analysis

4.2.1 Measures

Three measures are reported for the experimental outcomes on the test dataset, each one characterized by the mean and standard deviation in the results section:

Mean Squared Error Equivalent to the training criterion de ned in (2.1.4). The MSE does not capture predictive uncertainty and is thus used only as a minor comparison to observe the training success of the networks.

Negative log predictive denstiy The negative log predictive density (nlpd) is a strictly proper scoring rule [72][73] commonly used to compare the quality of predictive uncertainty between di erent models[74][75][76].
It is pointwise computed as de ned in equation (4.2.1):

$$L = \frac{1}{n} \sum_{i} -\log(p(y_i|x_i)) \tag{4.2.1}$$

[1]Note that the priors were the same for all the models directly derived from the random initialization ensemble, i.e., the Shu e and the bootstrap ensemble. Priors were not available for the ensembles relying on self-ensembling through time since the empirical prior can only be computed for ensembles in which a pre-initialization of the models takes place. See more on priors in this work in the Discussion 6.2

In our case of a regression with assumed normally distributed errors, the terms are:

$$L_i = \frac{1}{2}\frac{(\hat{y}_i - y_i)^2}{\sigma_i} + log(\sigma_i)(+C) \tag{4.2.2}$$

where $\hat{y}_i$ is the ensembles predictive mean as de ned in (2.3.2), σ_i is the ensembles predictive uncertainty as de ned in (2.3.3), y_i is the actual target value and C is a constant. To avoid numerical instabilities from very low uncertainty, we added 0:0001 during the computation.
Like most likelihood measures, nlpd can only be used to compare models' performance on the same dataset and are not transferable.

Correlation between uncertainty and error Correlation Between Error and Uncertainty (cobeau) is simply put Pearson's Correlation Coe cient computed between the vector of errors and the vector of uncertainty a model produces. It is a measure of how much linear correlation the models uncertainty has with the error the model makes[77].

$$\text{cobeau} = \frac{\sum_i (\sigma_i - \bar{\sigma})(\epsilon_i - \bar{\epsilon})}{\sqrt{\sum_i (\sigma_i - \bar{\sigma})^2}\sqrt{\sum_i (\epsilon_i - \bar{\epsilon})^2}} \tag{4.2.3}$$

Where $\bar{\epsilon}$ and $\bar{\sigma}$ respectively denote the mean and the standard deviation of the error ϵ and of the uncertainty σ and ϵ_i and σ_i the i-th entry in the vector of error and standard deviation.

P-values corresponding to cobeau We also report the p-values indicating the probability of observing values of this e ect or higher by chance[2].

4.2.2 Outlier Detection

Unfortunately, Neural Networks are susceptible to bad initialization values, which lead to numerical instabilities or vanishing gradients [79]. In order to lter out models for which the initialization was bad enough for them to generally not be considered representative of the ensemble's behavior under good conditions, the instances where this happens need to be removed from the experimental data. Fortunately, we can query the dataset for outliers based on a metric largely independent of the performance of the uncertainty of the Network, solving the dilemma of removing outliers based on the score the experiment is meant to measure. The MSE, which is available for each model, was used to identify and remove o ending instances from the analysis by computing a z-score over the error and removing models that exceed three standard deviations.

[2] The application of p-values has recently gone through a time of criticism because they were mis- and abused, especially in the social sciences, see, e.g., [78]

Chapter 5

Experiments

In this section, we report the outcomes of the experiments carried out on each dataset. Shown are the mean squared error, negative log predictive density, the correlation between error and uncertainty, and the corresponding p-value of each ensemble as well as baseline models as available. Reported are each measure with mean and standard deviation for each experiment with a di erent randomized train-test split after removing the outliers. Lower values are better for mse, nlpd, and the p-value, where larger values are better for cobeau. Visualizations report a credible interval of four standard deviations.

5.1 Small synthetic dataset

Results

Figure 5.1 shows the ensembles' behavior after 200 epochs of training. All models exhibit higher uncertainty in the out of sample regions than over the training data, with only the random initialization ensemble and the shu e ensemble exhibiting negligible uncertainty over the training data.

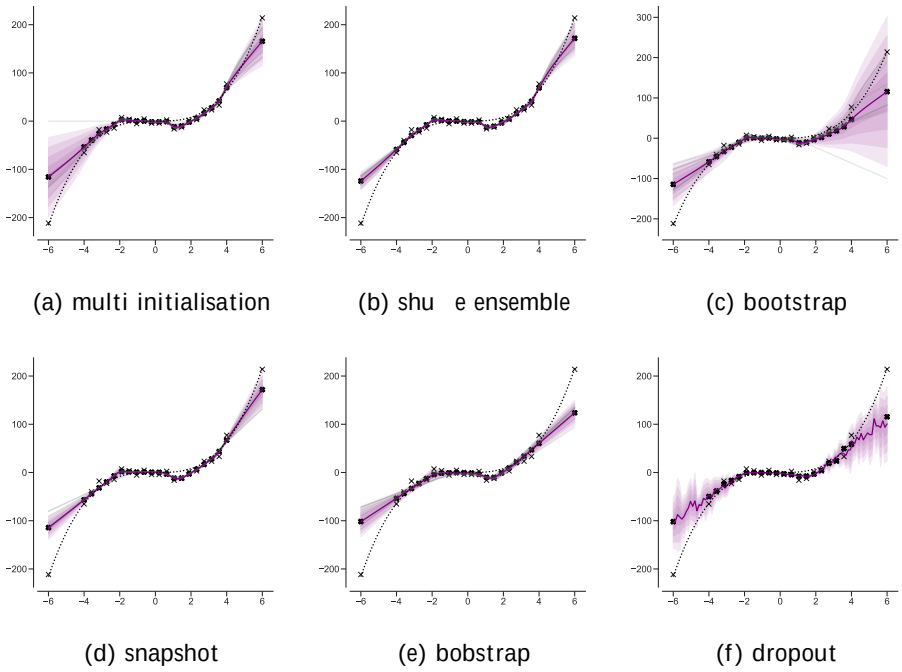
Analysis

All models seem to perform adequately and as one would expect for out of sample uncertainty. Interestingly, [31] reports signi cantly overcon dent estimations from multi-model ensembles on a similar dataset. However, note that we did not aim to perform a 1:1 comparison with their work, di erences include the number of epochs as well as architecture. The original literature works with di erent hyperparameters for learning rate decay and subsequently only trains their model for 40 epochs. In our experiments, this leads to severe underperformance. A visualization of the same models after 40 and 1000 epochs respectively can be found in gure C.1 and C.2.
The outcomes of these experiments seem to support the claim that comparing di erent ways of generating a predictive distribution carries the risk of comparing suboptimally optimized architectures rather than the actual way of generating distributions.

5.2 Large synthetic dataset

Results

Figure 5.2 shows a prototypical visualisation of the models' performance on the larger synthetic data set after 500 epochs. The models based on standard ensembling exhibit



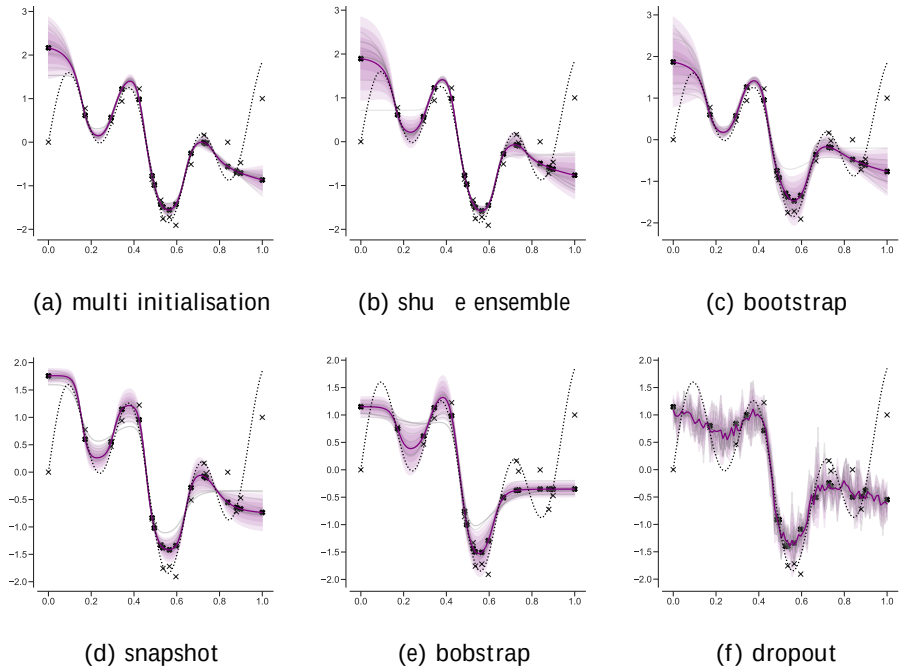
(a) multi initialisation (b) shu e ensemble (c) bootstrap

(d) snapshot (e) bobstrap (f) dropout

Figure 5.1: *Uncertainty typical for di erent ensembles on small synthetic dataset. The x-axis shows the x value, the y-axis shows the target. The dotted line represents the ground truth, small dots indicate training data, big dots are the models prediction. The predictive mean of the model is given by the continuous line, the models uncertainty in four standard deviations is given by the purple shade.*

stronger uncertainty in out of sample regions compared to the models based on snapshots, with the dropout model seemingly exhibiting similar levels of uncertainty at every region of the data. All models seem to capture the true distribution reasonably well in areas where training data was given, interestingly in the snapshot and the dropout models the regularizing e ect seems most substantial, preventing them from approximating the generating function as well as the other models in this particular train-test-split.
Table 5.1 reports the experimental outcomes for the synthetic data set without out of sample, table 5.2 reports the outcomes including out of sample data points.

No out of sample data As we can see, the mse of all models is signi cantly lower than that of the priors and the minimally informed model consisting of mean and std of the training data, with the bobstrap and the Dropout performing worst and the multi-model ensembles performing best. Unsurprisingly, the models are outperformed by the generating function in this respect.
When looking at the nlpd, the multi-model ensembles perform best, outperforming the generating function. They are followed by the bobstrap, the dropout, and the snapshot ensemble performing worst of the six. Notably, all models perform better than the priors and the empirical mean and standard deviation of the training data.
The cobeau is minimal for all models; the p-values indicate that the correlations could very well be chance.



(a) multi initialisation (b) shu e ensemble (c) bootstrap

(d) snapshot (e) bobstrap (f) dropout

Figure 5.2: *Uncertainty typical for di erent ensembles on larger synthetic dataset. The x-axis shows the x value, the y-axis shows the target. The dotted line represents the ground truth, small dots indicate training data, big dots are the models prediction. The predictive mean of the model is given by the continuous line, the models uncertainty in four standard deviations is given by the purple shade.*

Out of sample data included When adding the oos data, as the **mse** increases the gap in performance between the models closes.
The **nlpd** of all models becomes signi cantly worse, most notable for the snapshot ensemble, which is now outperformed even by the empirical metrics derived from the training data set. None of the models outperform the generating function after the inclusion of the oos data. The Bootstrap ensemble, previously very similar to the other multi-model ensembles, now di erentiates itself with the best value.
Regarding the **cobeau**, the three multi-model ensembles now report high correlations between their error and their uncertainty, with p-values indicating that a random observation is unlikely to exhibit this behavior while the other three models' error seems uncorrelated to their uncertainty.

Analysis

A salient comparison stems from how the models' relative performance changes with and without out of sample data. The **mse** of the models increases in similar ways while noticeably closing the gap between the ensembles. The **nlpd** rating of the models changes, with the bootstrap dominating the other ensembles on this measure after the inclusion of the new data points whereas it performed comparatively well but not outstanding compared to the others without those data. The snapshot model, arguably the weakest in both cases, even loses to the empirical metrics after the inclusion of the oos samples. The **cobeau** becomes large for the three multi-model ensembles with the extra data while being negligible without it. Compared to the nlpd, however, the ranking between

	errors		nlpd		cobeau		p-val	
VanillaEnsemble	0.29	0.07	-1.47	0.68	0.19	0.31	0.43	0.36
Shu eEnsemble	0.29	0.07	-1.47	0.68	0.19	0.31	0.43	0.36
BootstrapEnsemble	0.29	0.07	-1.43	0.33	0.09	0.3	0.45	0.33
snapshotModel	0.31	0.08	-0.81	1.3	0.12	0.27	0.44	0.26
BobstrapEnsemble	0.34	0.09	-0.91	0.86	-0.08	0.22	0.49	0.23
DropoutModel	0.36	0.08	-0.88	0.42	-0.08	0.19	0.63	0.28
multi initialisation prior	0.93	0.01	1.37	0.41	0.05	0.29	0.39	0.27
Dropout Model prior	0.94	0.03	16.12	2.76	0.04	0.2	0.55	0.25
train set mean/std	0.93		0.57		-		-	
generating function	0.26		-1.03		-		-	

Table 5.1: Large synthetic dataset results, no out of sample data, means standard deviations

	errors		nlpd		cobeau		p-vas	
VanillaEnsemble	0.44	0.05	-0.78	0.56	0.66	0.16	0.02	0.06
Shu eEnsemble	0.44	0.05	-0.78	0.56	0.66	0.16	0.02	0.06
BootstrapEnsemble	0.44	0.06	-0.9	0.24	0.63	0.13	0.02	0.02
snapshotModel	0.45	0.06	0.81	2.32	0.4	0.23	0.18	0.24
BobstrapEnsemble	0.45	0.07	-0.23	0.72	0.17	0.29	0.36	0.31
DropoutModel	0.46	0.06	-0.56	0.32	0.04	0.21	0.51	0.26
multi initialisation prior	0.89	0.0	1.18	0.37	0.12	0.26	0.41	0.3
Dropout Model prior	0.9	0.02	15.23	2.59	0.06	0.23	0.54	0.34
train set mean/std	0.89		0.53		-		-	
generating function	0.28		-0.98		-		-	

Table 5.2: Large synthetic dataset results, out of sample data present, means standard deviations

the two measures is reversed with the oos data: where the bootstrap is the weakest with regards to cobeau, it exhibits the strongest value for nlpd. This seems to indicate two things: On the one hand, cobeau seems to be a good indicator of how well a model treats out of sample data, on the other, it shows that a model can perform relatively better with regards to nlpd while having relatively weaker cobeau.

5.3 Housing data

Results

Figure 5.3 shows a prototypical visualisation of the models' performance on the housing data set with four standard deviations after 300 epochs of training.
Table 5.3 shows the outcomes for the real-world housing data. Since the data generating process is not available in this case, the comparison is omitted.
The mse of the ensembles is similar, outperforming the priors and empirical metrics by a wide margin.
The nlpd of the bobstrap is the best, followed by the multi-model ensembles (bootstrap slightly worse than the other two) and the dropout. The snapshot ensemble shows bad performance in this measure, being outperformed by both the empirical metrics as well as the multi-model prior.

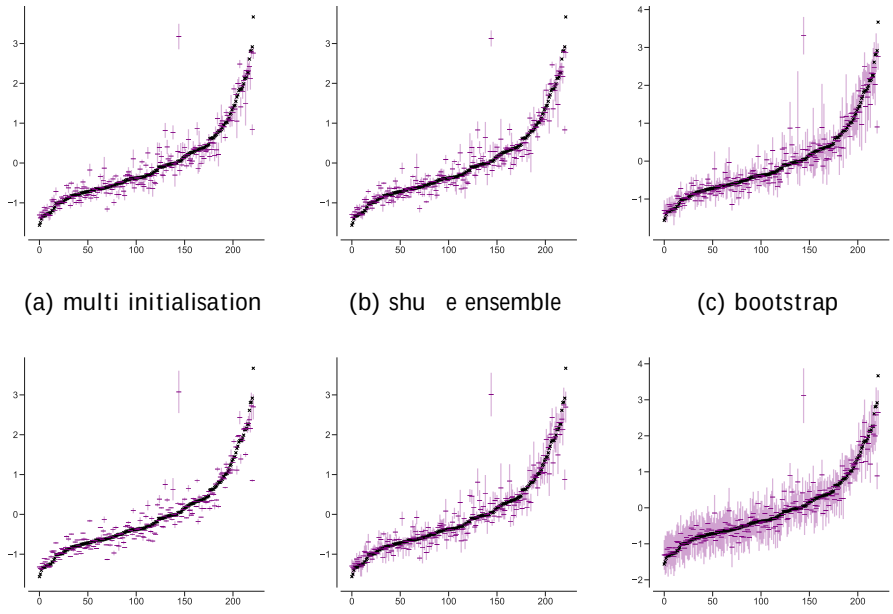
The cobeau of the snapshot ensemble is the highest; next, are the three multi-model ensembles with bootstrap and shu e ensemble scoring slightly higher than the third one. The Bobstrap, while slightly outperformed, still shows a strong correlation between error and uncertainty. The dropout ensemble performs poorly with regards to this measure, with no correlation to speak of.

Analysis

Visual inspection seems to indicate similar performance concerning mse with very different levels of con dence. The Dropout model exhibits large uncertainty almost everywhere; the snapshot ensemble seems overly con dent. Given how the uncertainty is generated, this can be blamed on small changes through the latter epochs. Indeed, training the model on fewer epochs or adding older snapshots to the prediction seems a remedy to this issue to some extent. This di erence to the other models points towards the need for a framework that enables a fair comparison. The experimental results seem to reinforce our ndings from the previous experiments about the orthogonality of nlpd and cobeau. Especially when taking into consideration the very similar values for mse: While the multi-model ensembles are in the mid- eld for both cobeau and nlpd, the snapshot ensemble exhibits both the best score for cobeau and simultaneously the worst for nlpd, being outperformed by the empirical metrics as well as the multi-model prior. Also, the Dropout, which is relatively close to the well-performing models when it comes to nlpd shows no correlation between its error and its uncertainty.
Compared to the synthetic data set, the bobstrap and the snapshot models gain with regards to cobeau, to the point where they are now among the better performers. A possible explanation for this divergence could lie in the way these models obtain their predictive uncertainty, which might work better on more complex data, but this is in the area of speculation.

	errors		nlpd		cobeau		p-val	
VanillaEnsemble	0.21	0.01	-2.02	0.29	0.37	0.14	0.0	0.0
Shu eEnsemble	0.21	0.01	-2.02	0.29	0.38	0.13	0.0	0.0
BootstrapEnsemble	0.21	0.01	-2.04	0.08	0.39	0.12	0.0	0.0
snapshotModel	0.21	0.01	1.27	2.3	0.44	0.14	0.0	0.02
BobstrapEnsemble	0.21	0.01	-2.14	0.12	0.33	0.1	0.0	0.0
DropoutModel	0.22	0.01	-1.73	0.09	0.03	0.08	0.51	0.35
multi initialisation prior	0.78	0.01	1.16	0.24	-0.02	0.08	0.34	0.22
Dropout Model prior	0.78	0.03	13.85	1.82	0.04	0.08	0.39	0.3
train set mean/std	0.77		0.47		-		-	

Table 5.3: Housing data results, means standard deviations



(a) multi initialisation (b) shu e ensemble (c) bootstrap

(d) snapshot (e) bobstrap (f) dropout

Figure 5.3: *Uncertainty typical for di erent ensembles on the housing dataset. Black dots indicate the true test data; the horizontal lines indicate the model's prediction. The horizontal lines indicate the credible interval of four standard deviations. The y-axis shows the target value; the x-axis indicates the sample. Two amendments were made solely for the visualization given the high dimensionality of the data as well as its origin in real-life data: the x-axis shows an indicator of the sample since the original x values have a dimensionality of 37; the data has been sorted by ascending y value to aid interpretability.*

Chapter 6

Discussion

Following, we will provide some discussion on the results of the practical evaluation, point out some limitations of the experimental setup, and look towards possible future research directions.

6.1 Discussion of results

In this section, we will discuss the main results from the practical evaluation of the six ensembles as well as some theoretical observations.

Practical Results

We draw three major conclusions from the practical results:

Measures A measure of the correlation of uncertainty and error seems to complement standard measures of likelihood such as nlpd. They capture different{but equally important{qualities of the predictive distribution. While the nlpd focusses on the overall fit with regards to error and the scale of uncertainty, measures such as cobeau indicate how reliable the uncertainty is concerning predicting an actual error the model is going to make.

Ensembles The ensembles perform adequately on the data sets with different strengths and weaknesses and no clear winner. While the mse of most models seem close to each other, the dropout model generally showed the weakest performance of the six, if not by a lot. The reason for this could be that while dropout seems like a minimal adjustment of architecture{only affecting 5% of many hundred nodes{it could already be enough to warrant, e.g., larger training cycles due to information lost through the dropout. On the other hand, the snapshot ensemble seems to be trained for too many epochs, which leads to large values for nlpd. Another interesting observation in this regard is the apparent failure of the Dropout model to capture the correlation between error and uncertainty in cases where the other models seemed to show medium-strong performance. In relative accordance with [31], the multi-model ensembles seem to perform most consistently over the synthetic and real-life data.

Experimental setup The comparison of predictive distributions in an experimental setup is a complicated task. As seen from the point above, even changes in architecture thought minuscule could have a significant effect on the performance of the models'

distribution. Additionally, these di erent architectures might a ect one measure while not changing the performance measured by others. Examples of this include the performance of the snapshot and the dropout ensembles, which exhibited bad nlpd and cobeau, respectively, while being competitive in regards to the mse. Additionally, some of our outcomes di er signi cantly from reports in the literature, even though we used a similar methodology. Notable is the performance of the ensembles on the small synthetic data set evaluated visually compared to the ndings of, e.g., [31], where our ensembles failed to converge in meaningful ways with similar choices for the number of epochs. These di erences, likely stemming from small adjustments made in the selection of hyperparameters, reinforce the authors' belief that comparing predictive distributions needs to be done in a very controlled environment. Even small di erences in training regimen can signi cantly skew the results towards one of the models being compared. We recommend the expenditure of resources to nd an experimental set up that facilitates comparability between di erent approaches to the generation of predictive distributions independently of architectures chosen to avoid false conclusions on the performance of di erent predictive distributions due to unfair comparisons.

6.2 Limitations

In the following section, we talk about some of this work's limitations and how they could be addressed in the future.

A note of the scale of uncertainty

There is ample literature on the calibration of uncertainty. [80] and his references are an exciting and useful place to start for the interested reader. For some of the models used in the comparison, the authors provide ways of stretching the uncertainty to proper scales, such as in [47]. However, due to the scope of this work as well as in the interest of comparability, we chose to only report the unscaled standard deviations over the distributions the ensembles themselves provided. For measures like the negative log predictive density, this could potentially lead to problems, given that they rely on the scale of the uncertainty measuring up to the scale of the error. Readers interested in applications of predictive distributions, such as Thompson sampling for reinforcement learning or other kinds of active learning or decision processes, are advised to read into the topic before trusting the methods described in this work.

Aleatoric vs. epistemic uncertainty

An important distinction in the research of predictive uncertainty is the di erence between *aleatoric* and *epistemic* uncertainty. While many de nitions exist, in essence, it is a question on which part of the uncertainty comes from the model being ill speci ed (e.g., through a faulty training regimen) vs. irreducible uncertainty stemming from noise in the data. While this work had to omit looking into this topic, several interesting approaches exist to estimate aleatoric vs. epistemic uncertainty, such as closely monitoring how well the distributions converge towards the generating function the noise added to the values given a large number of data points.

Models with intrinsically different architectures and the problem of optimization

A signi cant amount of thought went into this work's approach to forcing comparability between models. As a solution, we chose to omit models that could not be cast into

a shared architecture to ensure all models could be compared on a similar baseline of performance. While ensuring basic comparability of the models, this solution is unsatisfactory for several reasons. The most glaring one is that it is already suspected to introduce some bias via the xed number of epochs, which might over t the snapshot while under tting the dropout. Another signi cant limitation of this approach is having to drop interesting methods such as the formalisms introduced by [46] and [31] as they can't be guaranteed to show comparable performance in the restricted training regimen used for the models in this work. Research is needed into how to compare predictive distributions obtained via models on the complexity level of DNNs to avoid preferential treatment based on biased training regimens and parameter selections. A possible addition to the framework used in this work could include comparisons through time, which in our case could allow the snapshot model to conclude its training earlier while giving the Dropout model more time to develop fully.

Priors defined through ensembles

While research into prior distributions useable with BNNs and ensembles exists[1], this work consciously paid the issue minimal attention apart from using an empirical prior derived from multi-model ensembles and the dropout to give context to model performance. The prior of an ensemble of DNNs in our case is of the form $N(\ _{f\ g_m^M};\ _{f\ g_m^M})$, in which the location and scale parameters are computed as in equation (2.3.2) and (2.3.3). In the multi-model ensemble, this means it is dependent on draws from the distribution $U(\ \bar{p}_{\bar{k}};\ \bar{p}_{\bar{k}})$ as de ned in equation (3.3.1). In the case of the dropout, the spread is dependent on the dropout probability p that is applied to each neuron of a single draw from the same distribution, which de ned the location parameter. Finally, in the case of the snapshot ensemble, the prior is a degenerate distribution with location dependent on a single draw where $_{f\ g_m^M}$ equals 0 because we only have one instance of the model unless we train it for enough epochs to trigger a snapshot. Visualizations of the priors for multi-model and dropout can be found in gure 4.2

6.3 Future research questions

This section contains thoughts and comments on topics not included in this work due to scope as well as possible future lines of research in the area of predictive distributions from DNNs.

Possible link between Predictive distribution and variational approximation to the data generating process

While the relationship between the predictive distribution de ned in equation (2.3.1) and the posterior predictive distribution derived via a proper BNN in equation (2.2.1) can quickly be challenged{e.g. by pointing out that with this kind of ensembling we do not get a posterior over the parameters of the ensemble{the distribution might be related to a di erent form of Bayesian inference. As we can see in formula (3.4.2), in our case of mse regression, the parameters of the trained models can be assumed to be approximations of parameters that we would nd by minimizing the KL divergence between the predictive distribution and the data generating process. This might imply a link between the predictive distribution of ensembles and a variational approximation of the empirical distribution of the training data through a family of distributions whos

[1] see, e.g., [28] for an overview and [81] for a recent publication on the topic

parameters are de ned by the ensemble architecture. If this link indeed holds, ensembles of DNNs can, at least in this one scenario, be seen as proper Bayesian. This would give their already widespread practical use some additional theoretical basis[2]. However, some more research is necessary to probe this intuition for validity.

Recovering BNNs from snapshot models

While the argumentation above holds for general ensembles of DNNs, indicating that they can usually not be used as proper BNNs due to the lack of an explicit distribution over parameters, this might not be true for the snapshot model.
In the usual ensembling process, M independent networks are trained. Because their di erent initialization and the various e ects of stochastic gradient descent on their nodes, it is not possible to compare the values of corresponding neurons to obtain a distribution over parameters (neuron 1 in model 1 might respond strongly to a feature whereas neuron 1 in model 2 might react strongly to a di erent feature{thus averaging over them would not lead to improved performance but would likely be detrimental to the inference process).
However, in the snapshot ensembles, as trained in this work, the parameters of the different snapshots are not trained entirely independently, with di erent values quantifying the change through time for each parameter rather than an independently trained and thus incomparable DNN. This might open up the possibility of recovering an approximate posterior prediction over the parameters in this case.

Predictive power of the spread of the predictive distribution on the prediction error

The practical evaluation shows how a measure of the correlation between the predictive uncertainty on the model error, such as cobeau, can highlight di erent characteristics of a model compared to more conventional measures such as nlpd. More research is necessary to evaluate the quality of the measure and possibly derive a more robust measure based on a similar principle that is more sensitive to in-sample vs. out-of-sample data points.

Different architectures

Our base model architecture was found using grid-search with the average mse through several iterations as an optimization objective. While this approach was chosen to avoid interference with the other measures, a slightly di erent hyperparameter setup could be thought of. For example, one that optimizes, e.g., for a large spread of the mse to optimize the diversity of the multi-model ensembles or uses an entirely di erent measure to nd an optimal architecture.

Proactive boosting through uncertainty estimates

Boosting is an approach in which an ensemble is iteratively grown by training weak learners and re-weighting the input data by the error of the previous iteration[60].
In the presence of a predictive distribution, it is possible to conceive of an algorithm that uses information about its uncertainty to weight training examples. For example, a data point with low uncertainty but a large error could lead to a stronger adjustment in weights than a data point with low uncertainty and low error. Conversely, high

[2] Not limited to arti cial Neural Networks, either. It might have exciting implications for neuroscience as well.

uncertainty could indicate a datapoint carrying high educational value for the model and might thus be learned with a learning rate upscaled by high uncertainty.

Transfer of findings to classification

It is common for publications on BNNs to talk about regression and classi cation both. This work's scope, however, was limited to regression problems. While some of the theoretical assumptions will have to be overhauled for classi cation, the basic set up seems compatible with the problem of classi cation.

Predictive Distributions in Reinforcement learning

One area in which predictive uncertainty is valuable is in reinforcement learning. Knowing where a model is and is not con dent about its predictions enables an approach where the next action is chosen in relation. Thompson sampling, as used by [64] samples from the predictive distribution over di erent possible input values and then plays the one carrying the highest expected reward to solve the dilemma of exploration vs. exploitation. Other approaches, such as [39], aim to maximize the knowledge obtained over a particular domain. Thus they always play the action with the highest predictive uncertainty until it goes below a certain threshold. Classically, these approaches utilize Gaussian Processes, linear Bayesian Regressions, or ensembles of decision trees. While these approaches carry their bene ts such as explainability, in many situations using the more expressive class of models de ned by DNNs is preferred.

Chapter 7

Conclusion

We conclude the following insights from our work on predictive distributions derived via ensembling DNNs on regression problems:

Our theoretical and practical results point towards the predictive distribution derived via ensembling of DNNs being a valuable and computationally benign substitute to Bayesian Neural Networks for regression problems if used with caution and awareness of the models' strengths and weaknesses.

A measure for the predictive power of a model's uncertainty on its error adds quanti cation of out of sample behavior of models' predictive distribution. Our measure, COBEAU, could be used as a starting point in the development of such a metric.

Research into a standardized practical framework under which to compare di erent ways of obtaining a predictive distribution is likely to yield a `fairer' comparison between the methods. By avoiding arbitrary and non-representative advantages of one model architecture over another, better insights can be derived about which approach is preferable in speci c situations. The approach presented in this work is a rst step towards such a framework with obvious drawbacks when it comes to the type of model it is applicable for.

Appendix A

Technology used and Code Repository

Code repository

All code used in this work can be found at `https://github.com/wooohoooo/uncertainty_regression_pytorch`

Hard- and software used

This section contains a list of the hard- and software used in this project.

1. models were trained on a Dell XPS 13 with i7 CPU and 16 gb RAM, model of 2019.
2. programs written in python [82]
3. neural nets in pytorch [57]
4. train test split and other experimental approaches scikit learn[83]
5. scipy stats for outlier detection and the computation of the cobeau [84]
6. pandas for general data processing [85]
7. mathematical functions not found in pytorch: NumPy [86]
8. matplotlib and seaborn for visualisation [87], [88]
9. Code was developed in jupyter notebooks, [89]
10. LaTeX for writing [90]

Appendix B

Additional theoretical considerations

This appendix contains additional theoretical considerations and intuitions that had to be cut from the main body of the text due to scoping issues. In some cases, they only serve as the basis for an intuition and are not fully- edged mathematical arguments. Note that they are not mathematically rigorous since they are provided to aid an intuition rather than full proofs. The general assumptions and de nitions from the main text still apply.

Intractability intuition for bayes rule for models with large number of θ

This section provides a primer for big parameter spaces and why they make computation problematic.
This result is widely known, so we will focus on the main idea behind it: obtaining the model evidence by integrating out the parameters. A more thorough introduction to the topic can be found e.g., in [41] p128-132, section 5.5.

$$P(\ jD) = \frac{P(Xj\ )P(\ )}{P(X)} = \frac{P(X;\ )}{P(X)} \tag{B.0.1}$$

$$\text{for one theta: } P(X) = \int_{1} P(X;\ )d \tag{B.0.2}$$

$$\text{for two thetas: } P(X) = \int_{1}\int_{1} P(X;\ _1;\ _2)d\ _1 d\ _2 \tag{B.0.3}$$

$$\text{for N thetas: } P(X) = \int_{1}\int_{1} :::P(X;\ _1;\ _2;:::;\ _n)d\ _1 d\ _2 :::d\ _n \tag{B.0.4}$$

Derivation of recurrent definition of neural networks

[28] provides a derivation of DNNs from generalized linear models. The result, a recursive de nition of DNNs, is used in section 2.1.1. For convenience, we will retell the derivation in its basic steps here. We begin with a generalized linear model (glm) deriving its expected value for y_i on a new data point x_i from a combination of weights w, a bias variable b and a link function, g^{1}

$$\mathrm{E}[y_i|x_i] = g^{-1}(x_i w + b) \tag{B.0.5}$$

We then de ne this glm with an adaptive basis function by wrapping x in a function h depending on parameters , which perform some augmentation of the input data.

$$\mathrm{E}[y_i|x_i] = g^{-1}(h(x_i;)w + b) \tag{B.0.6}$$

From here we simply assume the data augmentation part to be another glm depending on a di erent set of weights and biases, recovering the de nition of a Neural Networks with one hidden layer

$$\mathrm{E}[y_i|x_i] = g^{-1}(h(x_i; W_1; b_1)w_2 + b_2) \tag{B.0.7}$$

where $h(x_i; W_1; b_1) = f(x_i W_1 + b_1)$
We can now keep stacking them recursively, arriving at

$$\mathrm{E}[y_i|x_i] = g^{-1}(h_L(x_i; \{W_l\}_{l=1}^{L}; \{b_l\}_{l=1}^{L})W_{L+1} + b_{L+1}) \tag{B.0.8}$$

where

$$h_l(x_i; \{W_j\}_{j=1}^{l}; \{b_j\}_{j=1}^{l}) = f_l(h_{l-1}(x_i; \{W_j\}_{j=1}^{l-1}; \{b_j\}_{j=1}^{l-1})W_l + b_l) \tag{B.0.9}$$

with recursion stop at $l = 0$: $h_0(x_i) = x_i$.

Parameters obtained by minimizing KL divergence through mse for large data sets in regression

We provide an intuition based on argumentation found in [41] or [42]. First, we show that the minimizing mse yields the same parameters as maximum likelihood estimation (MLE) in a gaussian model, then we show the link between MLE and D_{KL}.

Minimizing the Mean Squared Error as Maximum likelihood estimation for the parameters in a Gaussian model

Assume the Data is generated by a model $y = f(x|w) + $, where $\mathcal{N}(0;)$. Assume is known, and the data points are conditionally iid given w. We are modelling the data with a model $\hat{y} = \hat{f}(x;) + \hat{}$. In step (B.0.11) we used the fact that the noise is normally distributed
the log-likelihood of the model is

$$\log p(y|x;) = \sum \log p(y_i|x_i;) \tag{B.0.10}$$

$$= \sum_{i=1}^{N} \log \mathcal{N}(y_i; \hat{f}(x;); ^2)) \tag{B.0.11}$$

$$= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2 ^2}} \exp\left(-\frac{(y_i - \hat{f}(x;))^2}{2 ^2}\right) \tag{B.0.12}$$

$$= \frac{N}{2}\log 2 ^2 - \sum_{i=1}^{N} \frac{(y_i - \hat{f}(x;))^2}{2 ^2}) \tag{B.0.13}$$

where N is the number of observations. In (B.0.15) all constant terms have been dropped from (B.0.13).

$$_{mle} = \text{argmax} \log p(yjx;) \tag{B.0.14}$$

$$= \text{argmax} \quad N \sum_{i=1}^{N} (y_i \quad f(x;)^2) \tag{B.0.15}$$

$$= \text{argmax} \frac{1}{N} \sum_{i=1}^{N} (y_i \quad f(x;)^2) \tag{B.0.16}$$

$$\tag{B.0.17}$$

This is equal to the loss function de ned in (2.1.4).

MLE maximization as approximation to KL minimization for sufficiently large N

Assume we want to nd the parameters by minimizing the KL divergence $_{minKL}$ = argmin $D_{KL}(\hat{p}_{data}jjp_{model})$ and by applying the maximum likelihood principle, $_{mle}$ = argmax and show that they converge towards the same values for large N.

$$_{minKL} = \text{argmin} D_{KL}(\hat{p}_{data}jjp_{model}) \tag{B.0.18}$$

$$= \text{argmin} E_{x \ \hat{p}_{data}}[\hat{p}_{data}(X) \quad \log p_{model}(xj)] \tag{B.0.19}$$

$$= \text{argmin} \quad E_{x \ \hat{p}_{data}}[\log p_{model}(xj)] \tag{B.0.20}$$

In the next step, we turn the argmin of the negative value into an argmax, and given that the data are iid, we exchange the Expected value with a sum with a limit 1:

$$_{minKL} = \text{argmax} \lim_{N \to 1} \frac{1}{N} log(p(x_i j) \tag{B.0.21}$$

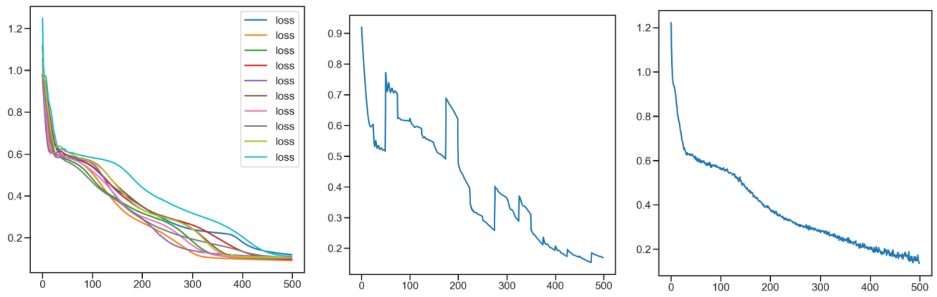
which, is the same as (B.0.14) up to a constant which does not change the choice of

Theoretical assessment of uncertatinty

Additionally, to the practical results in the main text, we provide some intuition into a possible interpretation of uncertainty stemming from theoretical considerations in combination with the outcomes of the experiments.

Multi model ensembles

The multi-model ensembles seem to derive their predictive uncertainty via a comparison of equally competent but di erently specialized members. A representative development over the individual models' losses, as can be seen in B.1a, supports this interpretation, showing the members converging to similar levels of competence. This seems coherent with the intuitive explanation for their better performance when using ensembles on datasets such as is common in competitions such as kaggle or the imagenet classi cation challenge. The uncertainty generated by this approach can point to data points that have very varying interpretations for each member, thus indicating them being hard to predict.



(a) typical ensemble loss (b) typical bobstrap loss (c) typical dropout loss

Figure B.1: typical training loss over epochs for di erent ensemble classes

Snapshot ensembles

The snapshot ensembles generate a distribution over their predictions from di erent points in time, with the bobstrap additionally changing the portion of the dataset that is being observed each time. It thus seems possible to pin down a source of their uncertainty as `surprise,' i.e., how much was learned in the passes over the data set since the last snapshot. Alternatively, it can be phrased as the di erence between the *prior* belief over the best hypothesis and the *posterior* belief. Figure B.1b shows the prototypical loss for a bobstrap ensemble with the switch of training data indicated by a spike in training loss followed by a region of a quick recovery. In situations where the old model predicts signi cantly di erent values to the newer one, the distribution has a large spread. In contrast, it is low when the old version of the model and the new agree. A similar notion of surprise is described, e.g., in [91] in the context of attention mechanisms. A combination of these notions might yield valuable insights into ensembling theory.

Dropout

The dropout ensemble derives its uncertainty from implicit ensembling[92]. Its derivation has an interesting contrast to multi-model ensembles, with the dropout utilizing slightly smaller architectures for their members due to the deactivation of nodes through dropout. Additionally, the parameters of the ensemble are shared in extreme fashion; each member subsamples several parameters from the ensemble space to make a prediction. This leads to two interesting observations:
While the multi-model ensembles are comprised of several specialized but equally competent members, the deactivation of certain nodes in the dropout inhibits such specialization. This is what dropout was originally conceived for - to regularize DNNs by removing the reliance on single neurons[66].
Instead, the dropout ensemble turns DNNs probabilistic by randomly sampling subnetworks. This behavior leads to the characteristic `wiggly' training loss shown in gure B.1c. Thus, the uncertainty derived from Dropout could be seen as a proper distribution over the networks likely spread of output values, with each sample being in uenced by a subset of beliefs the network carries.
Interestingly, this means that each sample could already be seen as a draw over the distribution of possible values (- whereas for the other ensembling methods, a single draw is determined by the network it was trained on - meaning we do not have information about how *likely* that particular draw is. It might be a freak outlier for all we know, and if we continuously resample from the members, it is likely to get an unfair share of attention). This is especially notable for methods using sampling from the nal

distribution over predictions such as Thompson Sampling (see, e.g., [93]), which can save a few computational cycles by simply using a forward pass as a draw.

Appendix C

Additional experimental outcomes

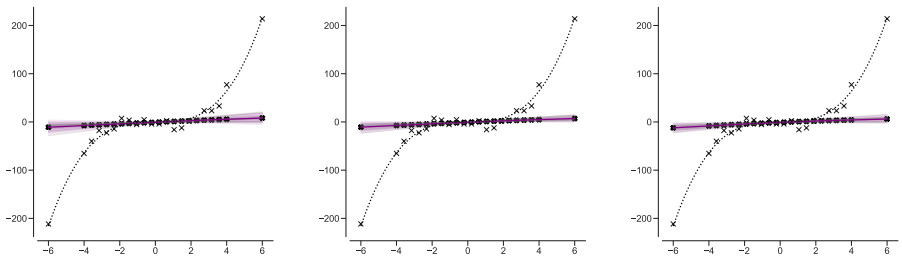
In addition to the experimental outcomes reported in 5, we performed other experiments that seemed interesting but were not strictly within the parameters set for the original research, mainly because they employ architectures not in line with the paradigm of nding the optimal architecture via grid-search optimized on mse.

Small Synthetic toy data set, 40 epochs

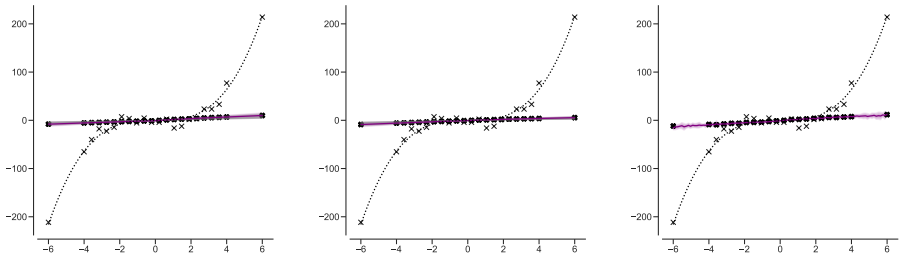
Our outcomes on the small synthetic data set were obtained via grid searching parameters and thus did not re ect the choice of parameters used in [31]. To obtain an intuition of the e ect on these parameter spaces, we tried to replicate similar results by training our network on 40 epochs. Our outcomes can be found in gure C.1. As we can see, our model severely under ts with this number of epochs.

Small synthetic dataset, 1000 epochs

Additionally, we trained the network for 1000 epochs to observe its behavior. The outcomes can be found in C.2. As we can see, for the multi-model, the bootstrap, and the dropout, this leads to even better t while keeping the predictive uncertainty relatively high.

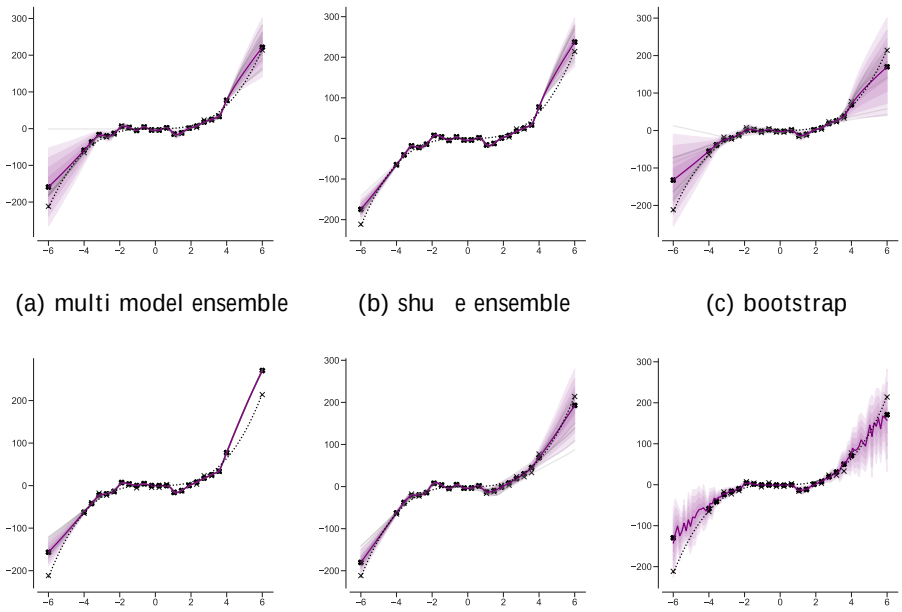


(a) multi model ensemble (b) shu e ensemble (c) bootstrap



(d) snapshot (e) bobstrap (f) dropout

Figure C.1: *Uncertainty typical for di erent ensembles on small synthetic dataset after 40 epochs. The x-axis shows the x value, the y-axis shows the target. The dotted line represents the ground truth, small dots indicate training data, big dots are the models prediction. The predictive mean of the model is given by the continuous line, the models uncertainty in four standard deviations is given by the purple shade.*



(a) multi model ensemble (b) shu e ensemble (c) bootstrap

(d) snapshot (e) bobstrap (f) dropout

Figure C.2: *Uncertainty typical for di erent ensembles on small synthetic dataset. The x-axis shows the x value, the y-axis shows the target. The dotted line represents the ground truth, small dots indicate training data, big dots are the models prediction. The predictive mean of the model is given by the continuous line, the models uncertainty in four standard deviations is given by the purple shade.*

Bibliography

[1] David Malako . *Bayes O ers a `New' Way to Make Sense of Numbers.(Bayesian statistics)*. Tech. rep. 1999.

[2] Carlos Riquelme, George Tucker, and Jasper Roland Snoek. \Deep Bayesian Bandits Showdown". In: 2018. URL: https://openreview.net/pdf?id=SyYe6k-CW.

[3] Sharon Bertsch McGrayne. *The theory that would not die : how Bayes' rule cracked the enigma code, hunted down Russian submarines, & emerged triumphant from two centuries of controversy*. Yale University Press, 2011, p. 320. ISBN: 9780300169690.

[4] Adrian E Raftery. \Choosing models for cross-classi cations". In: *American sociological review* 51.1 (1986), pp. 145{146.

[5] *A History of Bayes' Theorem - LessWrong 2.0*. URL: https://www.lesswrong.com/posts/RTt59BtFLqQbsSiqd/a-history-of-bayes-theorem (visited on 10/27/2019).

[6] F. Rosenblatt. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957. URL: https://books.google.nl/books?id=P_XGPgAACAAJ.

[7] Mikel Olazaran. *A Sociological Study of the O cial History of the Perceptrons Controversy*. DOI: 10.2307/285702. URL: https://www.jstor.org/stable/285702.

[8] Marvin Minsky and Seymour Papert. *Perceptrons; an introduction to computational geometry*. MIT Press, 1969, p. 258. ISBN: 9780262130431.

[9] G. Cybenko. *Approx_By_Superposition.Pdf*. 1989. DOI: 10.1007/BF02551274.

[10] Kurt Hornik. \Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251{257. ISSN: 08936080. DOI: 10.1016/0893-6080(91)90009-T.

[11] Alex Krizhevsky, Ilya Sutskever, and Geo rey E Hinton. *ImageNet Classi cation with Deep Convolutional Neural Networks*. Tech. rep. URL: http://code.google.com/p/cuda-convnet/.

[12] *From not working to neural networking - Technology*. URL: https://www.economist.com/special-report/2016/06/23/from-not-working-to-neural-networking (visited on 10/26/2019).

[13] Jacob Devlin et al. \BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[14] Zhenzhong Lan et al. \ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: (2019). arXiv: 1909.11942. URL: http://arxiv.org/abs/1909.11942.

[15] *wav2vec: Unsupervised Pre-training for Speech Recognition - Facebook Research*. URL: https://research.fb.com/publications/wav2vec-unsupervised-pre-training-for-speech-recognition/ (visited on 10/26/2019).

[16] *Understanding searches better than ever before*. URL: https://www.blog.google/products/search/search-language-understanding-bert (visited on 10/26/2019).

[17] Saining Xie et al. \Exploring Randomly Wired Neural Networks for Image Recognition". In: (2019). arXiv: 1904.01569. URL: http://arxiv.org/abs/1904.01569.

[18] Hans-Eric Jonsson and Kristian Nilsson. *A comparison of image and object level annotation performance of image recognition cloud services and custom Convolutional Neural Network models*. Tech. rep. 2019. URL: www.bth.se.

[19] Weicheng Kuo et al. \Expert-level detection of acute intracranial hemorrhage on head computed tomography using deep learning". In: *Proceedings of the National Academy of Sciences* (2019). URL: https://www.pnas.org/content/early/2019/10/15/1908021116.

[20] Tero Karras, Samuli Laine, and Timo Aila. \A Style-Based Generator Architecture for Generative Adversarial Networks". In: (2018). arXiv: 1812.04948. URL: http://arxiv.org/abs/1812.04948.

[21] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. \Waveglow: A Flow-based Generative Network for Speech Synthesis". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 3617{3621. ISBN: 9781479981311. DOI: 10.1109/ICASSP.2019.8683143. arXiv: 1811.00002.

[22] Hany Farid. \Creating, Weaponizing, and Detecting Deep Fakes". In: Santa Clara, CA: USENIX Association, Aug. 2019.

[23] David Silver et al. \Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529 (2016), pp. 484{503. URL: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[24] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. Tech. rep.

[25] Deepak Pathak et al. *Curiosity-driven Exploration by Self-supervised Prediction*. Tech. rep. arXiv: 1705.05363v1. URL: http://pathak22..

[26] Johan Kwisthout and Iris van Rooij. \Bridging the gap between theory and practice of approximate Bayesian inference". In: *Cognitive Systems Research* 24 (2013), pp. 2{8. ISSN: 13890417. DOI: 10.1016/j.cogsys.2012.12.008.

[27] *Deep Learning Is Not Good Enough, We Need Bayesian Deep Learning for Safe AI - Home*. URL: https://alexgkendall.com/computer{_}vision/bayesian{_}deep{_}learning{_}for{_}safe{_}ai/ (visited on 11/21/2019).

[28] Eric Thomas Nalisnick. *UC Irvine UC Irvine Electronic Theses and Dissertations Title On Priors for Bayesian Neural Networks*. Tech. rep. 2018. URL: https://escholarship.org/uc/item/1jq6z904.

[29] Diederik P Kingma and Max Welling. \Auto-Encoding Variational Bayes". In: (2013). arXiv: 1312.6114. URL: http://arxiv.org/abs/1312.6114.

[30] Diederik P. Kingma and Max Welling. \An Introduction to Variational Autoencoders". In: (2019). arXiv: 1906.02691. URL: http://arxiv.org/abs/1906.02691.

[31] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. \Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: (). URL: https://papers.nips.cc/paper/7219-simple-and-scalable-predictive-uncertainty-estimation-using-deep-ensembles.pdf.

[32] Yarin Gal and Zoubin Ghahramani. \Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: (2015). arXiv: 1506.02142. URL: http://arxiv.org/abs/1506.02142.

[33] *Ensembles and model combination*. Tech. rep. URL: http://www.inf.ed.ac.uk/teaching/courses/mlpr/2017/1.

[34] Je rey De Fauw et al. \Clinically applicable deep learning for diagnosis and referral in retinal disease". In: *Nature Medicine* 24.9 (2018), pp. 1342{1350. ISSN: 1546170X. DOI: 10.1038/s41591-018-0107-6.

[35] Leo Breiman. \Random Forests. transparencias". In: *Statistics* 45.1 (2001), pp. 1{33. ISSN: 08856125. DOI: 10.1023/A:1010933404324. arXiv: /dx.doi.org/10.1023{\%}2FA{\%}3A1010933404324 [http:]. URL: http://dx.doi.org/10.1023/A:1010933404324.

[36] Hugh A Chipman, Edward I George, and Robert E Mcculloch. \Bayesian Ensemble Learning". In: (). URL: http://papers.nips.cc/paper/3084-bayesian-ensemble-learning.pdf.

[37] *Bayesian Deep Learning*. URL: https://twiecki.io/blog/2016/06/01/bayesian-deep-learning/ (visited on 10/27/2019).

[38] Shengyang Sun et al. *FUNCTIONAL VARIATIONAL BAYESIAN NEURAL NETWORKS*. Tech. rep.

[39] D. Fan et al. \A robotic Intelligent Towing Tank for learning complex uid-structure dynamics". In: *Science Robotics* 4.36 (2019). URL: https://robotics.sciencemag.org/content/4/36/eaay5063.full.

[40] Bing Xu et al. \Empirical Evaluation of Recti ed Activations in Convolutional Network". In: *CoRR* abs/1505.00853 (2015). arXiv: 1505.00853. URL: http://arxiv.org/abs/1505.00853.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[42] Nando De Freitas. *CPSC540 Probabilistic linear prediction Probabilistic linear prediction and maximum likelihood*. Tech. rep. 2013.

[43] Stan Lipovetsky. \Analytical closed-form solution for binary logit regression by categorical predictors". In: *Journal of Applied Statistics* 42.1 (2015), pp. 37{49. ISSN: 13600532. DOI: 10.1080/02664763.2014.932760.

[44] David M. Blei, Alp Kucukelbir, and Jon D. McAuli e. \Variational Inference: A Review for Statisticians". In: (2016). DOI: 10.1080/01621459.2017.1285773. arXiv: 1601.00670. URL: http://arxiv.org/abs/1601.00670http://dx.doi.org/10.1080/01621459.2017.1285773.

[45] Geoorey E Hinton and Drew Van Camp. *Keeping Neural Networks Simple by Minimizing the Description Length of the Weights*. Tech. rep.

[46] Charles Blundell et al. \Weight Uncertainty in Neural Networks". In: (2015). arXiv: 1505.05424. URL: http://arxiv.org/abs/1505.05424.

[47] Yarin Gal. \Uncertainty in Deep Learning". In: (2016). URL: http://www.cs.ox.ac.uk/people/yarin.gal/website/thesis/thesis.pdf.

[48] Marcin B Tomczak and Richard E Turner. *Neural network ensembles and variational inference revisited*. Tech. rep. 2018.

[49] Alan E. Gelfand and Adrian F.M. Smith. \Sampling-based approaches to calculating marginal densities". In: *Journal of the American Statistical Association* 85.410 (1990), pp. 398{409. ISSN: 1537274X. DOI: 10.1080/01621459.1990.10476213.

[50] Josiah Willard. Gibbs. *Elementary Principles in Statistical Mechanics : Developed with Especial Reference to the Rational Foundation of Thermodynamics*. Cambridge University Press, 1902, p. 232. ISBN: 1108017029.

[51] Ludmila I Kuncheva. \Combining Pattern Classi ers Combining Pattern Classi ers Methods and Algorithms". In: (). URL: http://www.ccas.ru/voron/download/books/machlearn/kuncheva04combining.pdf.

[52] Kenji Kawaguchi. \Deep learning without poor local minima". In: *Advances in Neural Information Processing Systems*. Neural information processing systems foundation, 2016, pp. 586{594. arXiv: 1605.07110.

[53] Timur Garipov et al. *Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs*. Tech. rep. arXiv: 1802.10026v4. URL: https://github.com/timgaripov/dnn-mode-connectivity.

[54] Tiago M Fragoso and Francisco Louzada Neto. *Bayesian model averaging: A systematic review and conceptual classi cation **. Tech. rep. 2014. arXiv: 1509.08864v1.

[55] Tim Pearce et al. \Bayesian Inference with Anchored Ensembles of Neural Networks, and Application to Exploration in Reinforcement Learning". In: (2018). arXiv: 1805.11324. URL: http://arxiv.org/abs/1805.11324.

[56] Gao Huang et al. \Snapshot Ensembles: Train 1, get M for free". In: (2017). arXiv: 1704.00109. URL: http://arxiv.org/abs/1704.00109.

[57] Adam Paszke et al. \Automatic Di erentiation in PyTorch". In: *NIPS Autodi Workshop*. 2017.

[58] Diederik P. Kingma and Jimmy Ba. \Adam: A Method for Stochastic Optimization". In: (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[59] *pytorch/linear.py at master pytorch/pytorch*. URL: https://github.com/pytorch/pytorch/blob/master/torch/nn/modules/linear.py (visited on 10/23/2019).

[60] David Opitz and Richard Maclin. \Popular Ensemble Methods: An Empirical Study". In: *Journal of Arti cial Intelligence Research* 11 (1999), pp. 169{198. ISSN: 10769757. DOI: 10.1613/jair.614. arXiv: 1106.0257.

[61] Thomas G. Dietterich. \Ensemble methods in machine learning". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Arti cial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 1857 LNCS. 2000, pp. 1{15. ISBN: 3540677046. DOI: 10.1007/3-540-45014-9_1.

[62] B. Efron. \Bootstrap Methods: Another Look at the Jackknife". In: *The Annals of Statistics* 7.1 (1979), pp. 1{26. ISSN: 0090-5364. DOI: 10.1214/aos/1176344552.

[63] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Springer Series in Statistics The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Tech. rep.

[64] Dean Eckles and Maurits Kaptein. \Thompson sampling with the online bootstrap". In: (2014). arXiv: 1410.4009. URL: http://arxiv.org/abs/1410.4009.

[65] Dennis E. Taylor. *We are legion : (we are Bob)*, p. 299. ISBN: 9781680680584.

[66] Nitish Srivastava et al. *Dropout: A Simple Way to Prevent Neural Networks from Over tting*. Tech. rep. 2014, pp. 1929{1958.

[67] Jiri Hron, Alexander G. de G. Matthews, and Zoubin Ghahramani. \Variational Gaussian Dropout is not Bayesian". In: (2017). arXiv: 1711.02989. URL: http://arxiv.org/abs/1711.02989.

[68] Ian Osband and Google Deepmind. *Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout*. Tech. rep. arXiv: 1602.04621. URL: http://bayesiandeeplearning.org/2016/papers/BDL{_}4.pdf.

[69] Jose Miguel Hernandez-Lobato and Ryan Adams. \Probabilistic backpropagation for scalable learning of bayesian neural networks". In: *International Conference on Machine Learning*. 2015, pp. 1861{1869.

[70] Dean De Cock. *Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project*. Tech. rep. 3. 2011. URL: www.amstat.org/publications/jse/v19n3/decock.pdf.

[71] Yann LeCun et al. \E cient BackProp Lecture Notes in Computer Science". In: *Machine Learning and Knowledge Discovery in Databases*. Chapter 3. Springer Berlin Heidelberg, 2012, pp. 9{48. DOI: 10.1007/978-3-642-35289-8_3. URL: http://link.springer.com/10.1007/978-3-642-35289-8{_}3http://link.springer.com/content/pdf/10.1007{\%}2F978-3-642-35289-8{_}3.pdfhttp://link.springer.com/chapter/10.1007/978-3-642-35289-8{_}3http://link.springer.com/chapter/10.1007/978-3-642-35289-8{_}3.

[72] Tilmann Gneiting and Adrian E Raftery. \Strictly Proper Scoring Rules, Prediction, and Estimation". In: (). DOI: 10.1198/016214506000001437. URL: https://www.stat.washington.edu/raftery/Research/PDF/Gneiting2007jasa.pdf.

[73] Andrew Gelman, Jessica Hwang, and Aki Vehtari. *Understanding predictive information criteria for Bayesian models* *. Tech. rep. 2013.

[74] Joaquin Quiнonero-Candela et al. \Evaluating Predictive Uncertainty Challenge". In: (). URL: https://pdfs.semanticscholar.org/2e5a/fed6eb2ef1e360618b7b1d545b7616d750b8.pdf.

[75] John Geweke and Gianni Amisano. *Wo r k i n g Pa P e r S e r i e S n o 9 6 9 / n oV e M B e r 2 0 0 8 CoMParing and eValuating BayeSian PrediCtiVe diStriButionS of aSSet returnS*. Tech. rep. 2008. URL: http://www.ecb.europa.eu.

[76] Cesar Ojeda et al. *Variable Attention and Variable Noise: Forecasting User Activity*. Tech. rep.

[77] K Pearson. *Notes on Regression and Inheritance in the Case of Two Parents Proceedings of the Royal Society of London, 58, 240-242*. 1895.

[78] *It's time to talk about ditching statistical signi cance*. 2019. DOI: 10.1038/d41586-019-00874-8.

[79] Siddharth Krishna Kumar. *On weight initialization in deep neural networks*. Tech. rep. 2017. arXiv: 1704.08863v2.

[80] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. \Accurate Uncertainties for Deep Learning Using Calibrated Regression". In: (2018). arXiv: 1807.00263. URL: http://arxiv.org/abs/1807.00263.

[81] Danijar Hafner and Google Brain Dustin Tran Google Brain Timothy Lillicrap DeepMind Alex Irpan Google Brain James Davidson. *Noise Contrastive Priors for Functional Uncertainty*. Tech. rep. arXiv: 1807.09289v3.

[82] G. van Rossum. *Python tutorial*. Tech. rep. CS-R9526. Amsterdam: Centrum voor Wiskunde en Informatica (CWI), 1995.

[83] F. Pedregosa et al. \Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825{2830.

[84] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scienti c tools for Python*. [Online; accessed <today>]. 2001{. URL: http://www.scipy.org/.

[85] Wes McKinney. \Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference* (2010), p. 51.

[86] S. Chris Colbert Stefan van der Walt and Gael Varoquaux. \The NumPy Array: A Structure for E cient Numerical Computation". In: *Computing in Science Engineering* 13 (), p. 22.

[87] John D. Hunter. \Matplotlib: A 2D Graphics Environment". In: 9 (2007), p. 90.

[88] Michael Waskom et al. *mwaskom/seaborn: v0.8.1 (September 2017)*. Sept. 2017. DOI: 10.5281/zenodo.883859. URL: https://doi.org/10.5281/zenodo.883859.

[89] Thomas Kluyver et al. \Jupyter Notebooks { a publishing format for reproducible computational work ows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press. 2016, pp. 87 {90.

[90] Leslie. Lamport. *LaTex : a document preparation system*. Addison-Wesley, 1986, p. 242. ISBN: 0201157969.

[91] Pierre Baldi and Laurent Itti. \Of bits and wows: A Bayesian theory of surprise with applications to attention". In: *Neural Networks* 23.5 (2010), pp. 649{666. ISSN: 08936080. DOI: 10.1016/j.neunet.2009.12.007.

[92] Kazuyuki Hara, Daisuke Saitoh, and Hayaru Shouno. \Analysis of dropout learning regarded as ensemble learning". In: (2017). DOI: 10.1007/978-3-319-44781-0_9. arXiv: 1706.06859. URL: http://arxiv.org/abs/1706.06859http://dx.doi.org/10.1007/978-3-319-44781-0{_}9.

[93] Shipra Agrawal et al. \Analysis of Thompson Sampling for the Multi-armed Bandit Problem". In: 2326.39 (2012), pp. 1{39. URL: http://proceedings.mlr.press/v23/agrawal12/agrawal12.pdf.