Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Christian Nilsen Lehre

Quantifying Predictive Uncertainty in Artificial Neural Networks

With a Case Study from the Norwegian Oil and Gas Industry

Master's thesis in Industrial Mathematics
Supervisor: Gunnar Taraldsen
Co-supervisor: Peder Aursand, Bjarne Andre Grimstad
June 2021

NTNU
Norwegian University of Science and Technology

AkerBP

Christian Nilsen Lehre

Quantifying Predictive Uncertainty in Artificial Neural Networks

With a Case Study from the Norwegian Oil and Gas Industry

Master's thesis in Industrial Mathematics
Supervisor: Gunnar Taraldsen
Co-supervisor: Peder Aursand, Bjarne Andre Grimstad
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Abstract

Two methods for constructing Bayesian neural networks (BNNs), MC Dropout and SGVB, are implemented and applied to a real-world dataset provided by the Norwegian E&P company Aker BP. The dataset consists of borehole data gathered from 34 wells from offshore Norway, and the problem aims at predicting the acoustic log of S-waves based on geophysical measurements. The resulting BNNs can quantify the uncertainty in the models (epistemic) and the uncertainty in the data (aleatoric), making up the total predictive uncertainty. Moreover, the aleatoric uncertainty is modelled in a homoscedastic and heteroscedastic way, and the latter model is shown to consistently outperform the former for both methods. Furthermore, a qualitative analysis of the aleatoric uncertainty clearly shows that it captures uncertainty in the data at particular points in space.

All considered models are shown to accurately estimate the predictive uncertainty by using an analogy between the Bayesian credible interval and the frequentist coverage probability for a wide range of significance levels. Moreover, by neglecting the aleatoric component, the resulting uncertainty becomes highly under-estimated. Consequently, the majority of the predictive uncertainty is attributed to the data.

For the SGVB models, the epistemic uncertainty is shown to be reducible in terms of increasing the training set size and the hypothesis space representing model complexity. The epistemic uncertainty provided by the homoscedastic and heteroscedastic models steadily decrease when observing more data until converging to approximately the same values. In terms of model complexity, a single layer BNN is needed for explaining away the epistemic uncertainty, while a linear model does not suffice. By further increasing the model complexity, the epistemic uncertainty remains constant. On the contrary, the provided epistemic uncertainty by the MC Dropout models is not affected by the size of the training set. However, they are highly dependent on the dropout rate.

The MC Dropout models need careful tuning of a hyper-parameter to obtain proper uncertainty estimates, while the SGVB models are more flexible in terms of epistemic uncertainty. Consequently, we conclude that the SGVB method is superior to MC Dropout in quantifying the predictive uncertainty in artificial neural networks.

Extensions to the methods and analyses are proposed to obtain an even richer representation of the predictive uncertainty and to reduce inference time and inductive bias.

Sammendrag

To metoder for å konstruere Bayesianske nevrale nettverk, MC Dropout og SGVB, er implementert og anvendt på et reelt datasett levert av det norske E&P selskapet Aker BP. Datasettet består av brønndata hentet fra 34 brønner utenfor Norskekysten, og problemet består i å predikere den akustiske S-bølge loggen basert på geofysiske målinger. De resulterende nettverkene kan forklare usikkerheten i modellene (epistemisk) og usikkerheten i dataene (aleatorisk), som utgjør den totale prediktive usikkerheten. Den aleatoriske usikkerheten er modellert på en homoskedastisk og heteroskedastisk måte, og den sistnevnte modellen presterer konsekvent bedre enn den førstnevnte for begge metodene. Videre viser en kvalitativ analyse at den aleatoriske usikkerheten tydelig fanger usikkerheten i dataen.

Alle betraktede modeller oppnår gode estimater for den prediktive usikkerheten. Dette er vist numerisk ved å bruke sammenhengen mellom Bayesianske kredibilitetsintervaller of frekventistiske konfidensintervaller. Videre, ved å se vekk fra den aleatoriske komponenten blir den resulterende usikkerheten høyt underestimert. Følgelig tilskrives størstedelen av den prediktive usikkerheten til dataen.

For SGVB-modellene er den epistemiske usikkerheten reduserbar både når det gjelder å øke treningssettets størrelse og hypoteserommet som representerer modellkompleksiteten. Den epistemiske usikkerheten til de homoskedastiske og heteroskedastiske modellene avtar jevnt når de observerer mer treningsdata, til de konvergerer til omtrent de samme verdiene. Når det gjelder modellkompleksitet, trengs en BNN med ett skjult lag for å forklare vekk den epistemiske usikkerheten, mens en lineær modell ikke er tilstrekkelig. Ved ytterligere å øke modellkompleksiteten forblir den epistemiske usikkerheten tilnærmet konstant. Den epistemiske usikkerheten til MC Dropout-modellene påvirkes tilsynelatende ikke av størrelsen på treningssettet. De er imidlertid sterkt avhengige av hyper-parameteren som tilhører dropout-laget som brukes til å trene modellene.

MC Dropout-modellene trenger nøye justering av en hyper-parameter for å oppnå nøyaktige usikkerhetsestimater, mens SGVB-modellene er mer fleksible når det gjelder den epistemiske usikkerheten. Følgelig konkluderer vi med at SGVB er en bedre metode enn MC Dropout for å estimere den prediktive usikkerheten til kunstige nevrale nettverk.

Det foreslås utvidelser av metodene og analysene for å få en enda rikere fremstilling av den prediktive usikkerheten, og for å redusere regnekraft og forbedre generalisering av modellene.

Preface

This work has been carried out as a final part of my master's studies in Industrial Mathematics at the Norwegian University of Science and Technology (NTNU), and concludes my time as a student. Prior to my studies at NTNU, I graduated with a BSc. in Geophysics from the University of Bergen. For my master's project, I wanted to use my knowledge in geophysics, and I reached out to Aker BP to ask if they wanted to cooperate. The inquiry resulted in this thesis, where Aker BP provided me with a dataset based on geophysical measurements. During my exchange semester in Bologna, Italy, I participated in a course in Bayesian statistics. I was intrigued by the topic, and quickly found out that there is a lot of ongoing research in the intersection of Bayesian inference and another great interest of mine, namely artificial intelligence and deep learning. I reached out to professor Gunnar Taraldsen at the Department of Mathematical Sciences at NTNU with a proposal for a master's project about Bayesian neural networks, and he gladly accepted me as his student.

I want to thank my formal supervisor, Professor Gunnar Taraldsen, for allowing me to work on a topic that I genuinely find interesting and important.

Furthermore, I want to thank my co-supervisor, Peder Aursand (Aker BP), for our meetings and discussions during my master's semester. Peder has practically been my main supervisor and provided me with much insight into the problem provided by dataset and practicalities in machine learning. Even though Peder went on paternity leave early in the semester, we continued to have weekly discussion in the evenings. I am very grateful to have had Peder as my supervisor, and I really enjoyed our collaboration.

Last but certainly not least, I want to thank my other co-supervisor, Bjarne Grimstad (NTNU and Solution Seeker), for taking the time to supervise me in the field of Bayesian Neural Networks. Even though Bjarne had too much on his plate this semester, he took some time off to supervise me simply out of his own interest in the topic. Our meetings have been truly inspiring, and I owe him a debt of gratitude for helping me shape my thesis.

Christian N. Lehre
Trondheim, June 2021

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Machine learning models as deep neural networks aim to approximate a functional relationship between the input and output, describing a data-generating process through a joint probability distribution. This is traditionally done by learning point estimates for the parameters in the network through optimization. These networks are deterministic by nature and will not provide any measure of uncertainty in their output. By introducing prior distributions into the network's parameters and updating the priors during training, the network can learn probability distributions rather than point estimates for its parameters. This allows us to reason about uncertainty when using the networks to make predictions, which is essential in scenarios where models are being used as decision-support, in particular in safety-critical contexts like autonomous vehicles and medical diagnosis. The resulting probabilistic models are referred to as Bayesian neural networks. Traditional neural network models are known for their lack of interpretability, and one is often referring to them as black-box models. By extending the traditional neural networks and allowing them to reason about their predictive uncertainty, we are attempting to open up the black box and increase the interpretability of the models.

This thesis extends upon the work done in my specialization project, where two different methods for obtaining Bayesian neural networks were explored and applied to a toy dataset in a regression and classification setting. Consequently, some of the theory and methodology is based on the specialization project.

1.1 Outline

This chapter gives the reader an introduction to the history of Bayesian neural networks before describing the different sources of uncertainty we are concerned with within predictive modelling. Further on, two examples highlighting the importance of quantifying the predictive uncertainty in machine learning are given. In the last section of this chapter, we present the problem statement and motivate the problem.

The remaining part of this thesis is structured in the following way. Chapter 2 provides a theoretical background for machine learning and the accompanying sources of uncertainty. Further on, the model assumptions and corresponding diagnostic tools for a linear regression model are presented before providing the theory of neural networks relevant for this thesis. Next, we present the theoretical framework of Bayesian inference. The final section of the theory chapter presents Bayesian neural networks, where neural networks and Bayesian inference are united.

In Chapter 3, two methods for constructing Bayesian neural networks are presented before describing the different methods for solving the problems stated in Section 1.5.

The experimental setting is presented in Chapter 4, where a motivation behind the provided problem is presented, as well as an exploratory analysis of the data. Furthermore, a preliminary analysis based on a linear model is performed to justify the choice of models before presenting the chosen models and architectures.

In Chapter 5, we present the results of our analysis before discussing our findings in Chapter 6. The thesis is concluded in Chapter 7, where the main findings are summarized.

1.2 History of Bayesian Neural Networks

The earliest article describing and investigating a Bayesian neural network (BNN) dates back to 1987 [57], where the authors develop a statistical framework to reason about the generalization error of a neural network. The authors show that using a Euclidean loss function to train a neural network is statistically equivalent to performing maximum likelihood with respect to a Gaussian distribution over outputs of the network. Further on, they define a prior distribution of the network's parameters and show that inference could theoretically be performed using Bayes rule. A few years later, in 1990, some of the colleagues of the previous authors extended upon the idea in [57], where they suggested applying Laplace's method to approximate the posterior distribution of the parameters in a neural network [11].

A more extensive study of BNNs was carried out in 1992, where Mackay suggested using model evidence to perform model comparison [41], and obtained the results following the Laplace approximation in [11]. With a large number of experiments using different models and configurations, Mackay showed that model evidence is correlated with generalization error. Furthermore, he showed that model misspecification could lead to situations where model evidence is not indicative of model generalization [15].

In 1993, Hinton and Van Camp suggested using the information-theoretic minimum description length to penalize the amount of information in the parameters of a neural network as a way of regularization [25]. This is the first attempt at performing variational inference on the parameters of a neural network to approximate the corresponding posterior distributions. Variational inference will be described in greater detail in Section 2.4.1.

A few years later, in 1995, Neal developed a hybrid Markov Chain Monte Carlo (MCMC) method to perform approximate inference in BNNs, known as Hamiltonian Monte Carlo (HMC) [43]. The method revolves around generating samples from the posterior distribution of the neural network parameters that are otherwise difficult or intractable to compute. The proposed method was the first use of MCMC algorithms applied to the parameters of neural networks. Moreover, the work was the first to establish a link between Gaussian processes and Bayesian neural networks.

Much modern research is being done in the field of Bayesian neural networks, and the majority extends on the work done by Hinton and Van Camp on variational inference in [25].

In 2011, Graves attempted to resolve the computational challenges in training BNNs with variational inference. In his work [19] the intractable expected log-likelihood term in the loss function is approximated using Monte Carlo sampling, allowing the method to scale better. A few years later, in 2015, Blundell et al. extend on Graves method in conjunction with the reparametrization trick proposed by Kingma and Welling [33] in 2013. In their work [7] the expected log-likelihood is re-parametrized to allow for backpropagation through stochastic nodes in the network. However, the method is restricted to Gaussian distribution of the parameters of the network and increases the number of parameters to be learned during training. One of the methods for constructing BNNs in this thesis is based on the method proposed by Blundell et al. and will be described in more detail in Section 3.2.

In his thesis from 2016, Yarin Gal showed that one could perform approximate inference on the parameters of Bayesian neural networks simply by training the network with a stochastic regularization technique known as dropout, leaving dropout on when making predictions and perform multiple predictions for each instance [15]. The method was shown to be mathematically equivalent to variational inference in deep Gaussian processes and will be described in more detail in Section 3.1.

In the following year, another interesting approach for obtaining predictive uncertainty in deep learning was suggested. However, the method can not be seen as approximate Bayesian inference, unlike the above-mentioned methods. The outline of the method consists of training an ensemble of traditional neural networks, each having a different initialization of the parameters [38]. Having obtained an ensemble of neural networks, one can estimate the predictive uncertainty by using the sample variance of the predictions provided by the ensemble. Although the method consists of training multiple neural networks, the computational complexity is typically lower than training a Bayesian neural network using approximate inference [38].

To allow for a richer representation of the predictive uncertainty provided by Gal and Blundell et al. methods, we will extend the methods to distinguish between different sources of the predictive uncertainty. An explanation of the different sources of uncertainty is explained in the following section.

1.3 Sources of Uncertainty

When discussing uncertainty in the context of predictive modelling, we need to define and distinguish between two inherently different sources of such, namely *aleatoric* and *epistemic* uncertainty [1, 12, 27, 30, 53].

Aleatoric uncertainty, also referred to as statistical uncertainty, represents the randomness of the outcome of an experiment. This means that the data-generating process potentially consists of a stochastic component independent of the amount of information available. This source of uncertainty is also referred to as the irreducible uncertainty since it is not possible to reduce this type of uncertainty by obtaining more information [27], without changing the underlying system by which the experiment is performed [15]. Aleatoric uncertainty is present in nearly all data we gather due to variability in the obtained samples from the population we are modelling or simply due to stochastic measurement errors [44]. If we change how we collect the data, e.g. by improving the measurement precision, the aleatoric uncertainty can be reduced. However, having collected the data, there are no ways to reduce the uncertainty in the data. It is important to note that the aleatoric uncertainty is an inherent property of the data-generating process and not of the model trying to explain it. An example of aleatoric uncertainty is the uncertainty related to dealing a deck of cards. No matter how well we can model the experiment, there will always be some component of uncertainty involved due to the stochastic nature of the experiment.

On the contrary, the epistemic uncertainty component refers to the lack of knowledge in the data-generating process. This type of uncertainty is also referred to as systematic uncertainty and can be reduced by introducing more information about the process or system being modelled [27].

Let us look at an example of epistemic uncertainty. Say you have recently moved to Italy, and you have no idea how to speak the italian language. The uncertainty in the way of speaking Italian is thus very high. You attend a language course and get better and better by the day. You become less uncertain in speaking Italian, and you have become so by attending the language course. Here, you have used the language course as additional information about speaking the language, and you have reduced the uncertainty by attending the course.

The main distinction between aleatoric and epistemic uncertainty revolves around whether one can reduce the uncertainty based on obtaining more information. This pragmatic definition makes the distinction ambiguous and context-dependent, and one must be careful when distinguishing between the two.

1.4 Importance of Quantifying Predictive Uncertainty

It will be evident in Section 2.1.2 that there is much uncertainty attached to the process of training a machine learning model and making predictions. However, the majority of machine learning techniques we see today do not provide any measure of uncertainty.

When making predictions, we input an instance $\boldsymbol{x}$ similar to the instances we trained the model on[1], and the trained model $\hat{m}$ will output a point estimate of the target.

$$\hat{y} = \hat{m}(\boldsymbol{x}) \in \mathbb{R},$$

where $\hat{y}$ is the predicted target, $\boldsymbol{x}$ is the instance we are predicting and $\hat{m}$ the trained model.

Having a model that outputs a point estimate without a measure of uncertainty is problematic, particularly if the predictions from the machine learning model are being used as decision support in a safety-critical context.

Below are two hypothetical scenarios highlighting the importance of quantifying the predictive uncertainty in machine learning models applied in real-world situations.

Medical diagonosis

Imagine a medical diagnosis scenario in which a doctor uses a machine learning model to predict the presence of a cancerous tumour based on an examination. Here the target is $y \in \{0, 1\}$, corresponding to the presence (1) and absence (0) of the tumour, and the instance $\boldsymbol{x}_{\text{patient}}$ is based on the examination. Assume that the model has not been sufficiently trained on instances similar to the patient, $\boldsymbol{x}_{\text{patient}}$, so the prediction will intuitively not be very confident. Since the model outputs a point estimate based on the instance $\boldsymbol{x}_{\text{patient}}$, we cannot say anything about the predictive uncertainty of the model. Imagine now that the output of the model is $\hat{y}(\boldsymbol{x}_{\text{patient}}) = 1$, i.e. the model is saying that there is a presence of a cancerous tumor. However, the prediction is a false positive. The doctor is naive and puts the patient on chemotherapy immediately, even though the patient is perfectly healthy.

Drilling for oil

Consider now a petroleum engineer working for a company deciding whether or not to drill for oil in an area. The engineer has a background in computer science and relies heavily on machine learning models in his daily workflow. In the decision process, the engineer collects relevant data in the area they are considering. Having trained a machine learning model on the same type of data, the engineer uses the model to predict whether they can expect oil to be present in the area. As it turns out, the output of the model shows the presence of oil. The engineer is very confident in his model and decides that the company will drill for oil in the area, spending much money hoping that the profit will be even greater than the expenses related to drilling and operating the field. Unfortunately, the model output turned out to be a false positive, and the company wasted much money drilling the dry well.

[1] using the same set of explanatory variables

The above are two fictitious examples highlighting the importance of quantifying the uncertainty related to machine learning, especially in scenarios where the model output is used as decision support. In both hypothetical scenarios, the model provided false positive predictions, resulting in erroneous decisions.

Ideally, we would have a model that returns a measure of uncertainty associated with each prediction. This would make it possible to decide whether one should pass an instance to a human for a more thorough inspection or to trust the prediction provided by the model.

1.5 Problem Statement and Motivation

This master's thesis is a collaboration with the Norwegian E&P company Aker BP, which provides a dataset for solving a regression problem. The company is currently using boosted trees [14] and have not yet explored the use of deep learning for the particular problem.

Neural network models are very flexible and have been seen to perform well in numerous real-world scenarios [39]. However, the models fail to provide any measure of uncertainty, and their predictions tend to be overconfident [7]. Having a neural network model that can provide a measure of uncertainty is highly valuable, as one can reason about how reliable the corresponding predictions are.

This thesis aims to construct Bayesian neural networks that can reason about the uncertainty of its predictions. Furthermore, the goal is to decompose the predictive uncertainty into the two components described in Section 1.3, namely the aleatoric and epistemic uncertainty. The resulting uncertainty estimates will be properly evaluated, and the importance of including both uncertainty components will be investigated. To validate the aleatoric uncertainty estimates, we will perform a qualitative analysis of the data, where we investigate whether the aleatoric uncertainty responds to noise in the data. Moreover, the following research hypotheses are empirically tested.

i. The epistemic uncertainty is inversely proportional to the amount of data

ii. The epistemic uncertainty depends on the complexity of the model

The former hypothesis has previously been stated [12, 27, 30], but we have not seen any experimental nor theoretical justification. The latter hypothesis is concerned with model misspecification and is based on the intuition that a bigger hypothesis space has a greater chance of containing the optimal model. We will come back to the notion of uncertainty due to the amount of data and model misspecification in Section 2.1.2.

Chapter 2

Theory

This chapter provides the reader with a theoretical background in machine learning, linear regression and neural networks relevant to this thesis. Further on, the theoretical framework provided by Bayesian inference is presented, where we describe a method for performing approximate inference. Bayesian inference and deep learning are coupled together in the final section of this chapter, where we describe Bayesian neural networks.

2.1 Machine Learning

In this section, we give a brief introduction to the field of machine learning before pointing out the different sources of uncertainty one should be aware of when working with such.

2.1.1 Brief Introduction

Before looking into the different sources of uncertainty in machine learning, we will look at one of many definitions of machine learning, this one provided by Tom Mitchell [42].

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

In the definition above, we refer to data as the experience E. The set of tasks T represents the problem we have at hand, e.g. whether we are concerned with a regression or classification problem. The performance task P is referred to as the loss- or cost-function, measuring the ability of the machine learning algorithm to perform the given task.

Machine learning algorithms are categorized based on the type of data they are allowed to process during training. There are numerous variations of each type of algorithm, but they are commonly categorized into *supervised-*, *unsupervised-* and *reinforcement* learning [18]. In supervised learning, the algorithms are allowed to process labelled data, thus guiding the algorithm towards correct predictions during training. In unsupervised learning, the algorithms typically rely on some measure of similarity to cluster input without a labelled response[1]. Supervised-

[1]Note that there are other techniques of doing unsupervised learning, but clustering is the quintessential technique for doing so

and unsupervised machine learning is quite similar, with the difference being in the ability to observe the correct prediction during training or not. Quite different from these two classes of algorithms, we have reinforcement learning. Reinforcement learning is concerned with finding the optimal action to take in an environment to maximize a reward function. The algorithm iterates in a trial-and-error fashion, finding the action that yields the best possible reward[5]. An example application of reinforcement learning is training a computer how to play the game of go [51], a strategic chess-like board game.

We will further on consider the task of supervised learning, where the training consists of processing labelled data. The dataset can in this case be written as

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\} \in \mathcal{X} \times \mathcal{Y},$$

where $\mathcal{X}$ is the space of possible explantory variables, $\mathcal{Y}$ the space of possible target variables and n the size of the dataset.

In addition to data, the generic machine learning problem consists of three integral parts:

1. **Model class**: A set of models $\mathcal{M}$ defined over a parameter space Θ. The model class is typically referred to as the *hypothesis space*, consisting of possible hypothesis for explaining the data. Some of the parameters are not updated during training, and the user needs to specify these before training the model. These parameters are referred to as *hyper-parameters*. The remaining parameters are updated during training by optimizing a criterion using a suitable optimization method.

2. **Criterion**: A function to be optimized. In supervised learning, this consist of a scalar loss function that captures the discrepancy between the true and predicted target variable. The criterion depends on the parameters of the model.

3. **Optimization method**: A suitable method for optimizing the criterion and updating the trainable parameters of the model. In supervised learning, this would be a method to minimize the discrepancy between the true and predicted target variable. A typical choice is the gradient descent algorithm or one of its many varieties. The choice of optimization methods typically introduces new hyper-parameters to the model.

The key problem of any machine learning application is the ability to generalize to unseen data, i.e. data the algorithm has not processed during training.

A common practice is to split the dataset into a training set and a test set, where the test set is kept aside when training the model. Further on, if the model consists of hyper-parameters that can take different values, the training set is further split into another training set and a validation set. The goal is to find the optimal hyper-parameter configuration by optimizing a suitable performance measure on the validation set. When the optimal configuration of hyper-parameters is found, the model is typically re-trained on the entire training set, including the validation set. When performing the split, it is important to avoid flow of information between the training and test set. The test set is kapt away from the model during training, and is used to measure model performance on unseen data. If information from the test set is included in the training set, the performance of the resulting model is not representative to performance on unseen data. This situation is referred to as *data leakage*, and is very important to avoid.

The ability of the model to generalize to unseen data is estimated in terms of performance on the test set and is measured using suitable performance metrics. An important assumption about the training and test set is that the datasets are independent and identically distributed (i.i.d), [18], i.e.

$$p_{\text{train}}(x, y) \sim p_{\text{test}}(x, y) \sim p_{\text{data}}(x, y), \tag{2.1}$$

where p_{train} and p_{test} are the joint distributions representing the training- and test set, and p_{data} the distribution representing the data-generating process. The i.i.d assumption allows to reason about the ability of a trained model to generalize to unseen data, as the data is drawn independently from identical distributions. Say, for example, we train a forecasting model to classify whether the weather will be sunny or cloudy tomorrow. We train our model using a training set consisting of sunny and cloudy days only. If we then test the model on an instance representing a rainy day, we cannot infer anything about the model's ability to generalize in its task of classifying whether the weather will be sunny or cloudy. We refer to this situation as *out-of-distribution* (OOD) test-data. OOD test-data typically result in *inductive bias* [2], where the generalization ability of the model becomes poor because of varying distributions between the datasets or a badly chosen hypothesis space for the model. There are however different methods for dealing with inductive bias and relaxing the i.i.d assumption (2.1), e.g multi-task learning [9] and transfer learning [55]. We will, however, not go through these methods.

The concept of over and under-fitting in machine learning is vital and directly related to the capacity or equivalently the complexity of a machine learning model. Allowing for too complex models will potentially make the model interpolate all the samples in the training set, which is only a random sample of the distribution of the data-generating process being modelled. When applying an over-fitted model to unseen data, the performance will generally be poor. On the contrary, the concept of under-fitting applies to situations where the machine learning model is too rigid with respect to the data. In this setting, rigidity refers to the fact that the model is not able to capture the variability in the data. This will generally result in poor performance, both on the training set and unseen data. Over- and under-fitting is closely related to the concept *bias-variance trade-off* seen in statistical learning [23]. The bias-variance trade-off describes a trade-off between the variance and bias of a statistical learning model, where low bias is accompanied by high variance and vice versa. The trade-off originates from a decomposition of the expected test error into three fundamental components, namely the variance, the squared bias, and the irreducible error of the predictions on the test set [29].

A depiction of the bias-variance trade-off is seen in Figure 2.1 below. The figure shows a hypothetical scenario where the prediction error is plotted against an increasing model complexity. The training error is seen as the blue curve, while the red curve shows the test error, i.e. the error on unseen data. The optimal model complexity is marked as the vertical, dotted line.

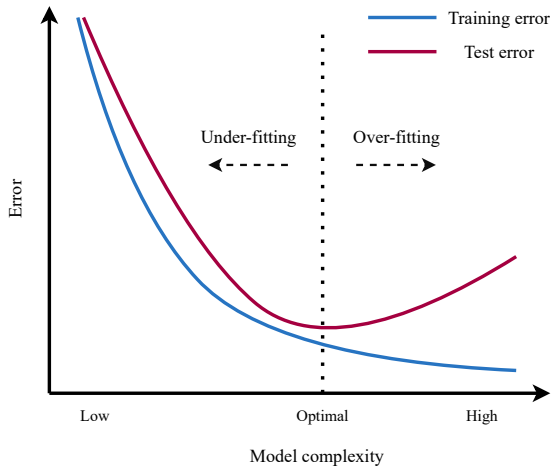


Figure 2.1: Bias-Variance trade-off. The training error is marked in blue, while the test error is red. The optimal complexity, shown by the vertical dotted line, minimize both the training and test error. Increasing the complexity further leads to over-fitting, while decreasing the complexity leads to under-fitting. Adapted from [23].

In the case of under-fitting, we are fitting a model that is too rigid to the data at hand, resulting in high bias. However, the variance is low. On the other side, i.e. in over-fitting, we are using a too complex model. This results in low bias, but the variance of the predictions is high. In the optimal model complexity, bias is traded off with variance to minimize the error on the unseen data. This will, in turn, result in the best generalization ability.

The capacity of a model is readily controlled, and there are numerous different techniques for doing so. These techniques are referred to as *regularization* techniques. Applying regularization to a machine learning model typically revolves around adding suitable penalty terms to the optimization objective, i.e. the loss function. In this way, we can say that the model's capacity, and thus the chance of either over or under-fitting, is determined by the model class and the loss function combined. There exist other types of regularization techniques as well, many of which are restricted to specific model classes and learning algorithms.

Given a hypothesis space $\mathcal{M}$ and a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, the supervised learning algorithm aims to infer a model $m^* \in \mathcal{M}$ minimizing the loss function over the training set, i.e.

$$m^* = \underset{m(\boldsymbol{\theta}) \in \mathcal{M}}{\operatorname{argmin}} \mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}), \tag{2.2}$$

where $\boldsymbol{y}$ are the ground-truth targets, and $\hat{\boldsymbol{y}} = m(X|\boldsymbol{\theta})$ the predicted targets from the instances X in the training set. The model m is defined in terms of a set of parameters $\boldsymbol{\theta} \in \Theta$, where Θ is the space of all the parameters defining the hypothesis space $\mathcal{M}$. The optimization in (2.2) is carried out by updating the parameters $\boldsymbol{\theta}$, thus finding the model that minimize the loss.

2.1.2 Sources of Uncertainty in Machine Learning

Application of machine learning models typically consists of using the trained models to make predictions in some context. Thus, when talking about uncertainty in machine learning, one is typically interested in the predictive uncertainty related to the model in the given context. This is particularly useful if the predictions are used to make decisions in the context where the model is applied. The predictions provided by a machine learning model constitute all the choices and approximations related to training the model and the process of collecting the data. Depending on the origin of the uncertainty, the total predictive uncertainty can be decomposed into aleatoric and epistemic uncertainty, which will constitute all the errors and uncertainties involved in the machine learning pipeline.

Let us consider a generic, supervised machine learning problem, where the goal is to map some functional dependency between an instance space $\mathcal{X}$ and a target space $\mathcal{Y}$. The mapping is typically performed by setting up a hypothesis space $\mathcal{M}(\Theta)$ containing different models that we believe explain the variability of the data well. Without complete knowledge of the perfect model, there will be uncertainty attached to setting up a proper hypothesis space for the problem at hand. However, this uncertainty can be reduced by obtaining more information about the process, e.g. by investigating whether or not a particular model is suitable for explaining the variability in the data. This uncertainty is referred to as *structural-* or *model uncertainty*, and falls under the umbrella of the epistemic component of the predictive uncertainty. In the case of a misspecified model, this type of uncertainty will be high. On the contrary, if the model is appropriately specified, there will be little model uncertainty. We can thus use the model uncertainty to investigate whether we have specified our model correctly.

The machine learning pipeline continues by finding the optimal set of parameters $\hat{\boldsymbol{\theta}} \in \Theta$, specified by the hypothesis space. The parameters are typically found by maximizing a likelihood function under some distributional assumptions of the functional form of the data-generating process. The rationale is to find the model in the hypothesis space that maximizes the probability of observing the data we have at hand. It is otherwise common to formulate the problem to minimize a suitable loss function[2] (2.2). The result of the optimization is a hypothesis $\hat{m}(\hat{\boldsymbol{\theta}}) \in \mathcal{M}(\Theta)$, which is an estimate of the best hypothesis $m^*(\boldsymbol{\theta^*})$ within the hypothesis space. This estimate strongly depends on the amount of training data [27], and there will typically be some discrepancy between the induced hypothesis $\hat{m}(\hat{\boldsymbol{\theta}})$ and the best hypothesis $m^*(\boldsymbol{\theta^*})$ within $\mathcal{M}(\Theta)$. We will refer to the uncertainty posed by this discrepancy as the *approximation uncertainty*. Due to its strong connection with the size of available training data, we can use the approximation uncertainty to assert whether or not it is appropriate to collect more training data. Similar to the model uncertainty, the approximation uncertainty is subsumed under the epistemic uncertainty component of the total predictive uncertainty.

[2]The loss function typically arise from the maximum likelihood formulation

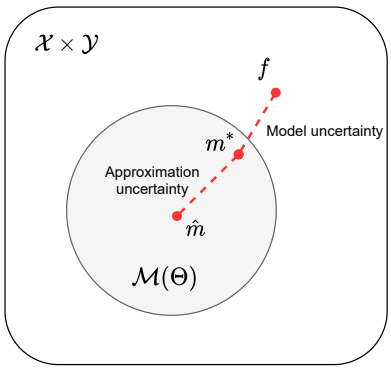


Figure 2.2: Illustration of the different components of the epistemic uncertainty seen in machine learning. the Cartesian product $\mathcal{X} \times \mathcal{Y}$ represents the data, and $\mathcal{M}(\Theta)$ the chosen hypothesis space. The induced model is $\hat{m}$, and the optimal model within the hypothesis space is m^*. The ground-truth model is represented by f. Adapted from [27].

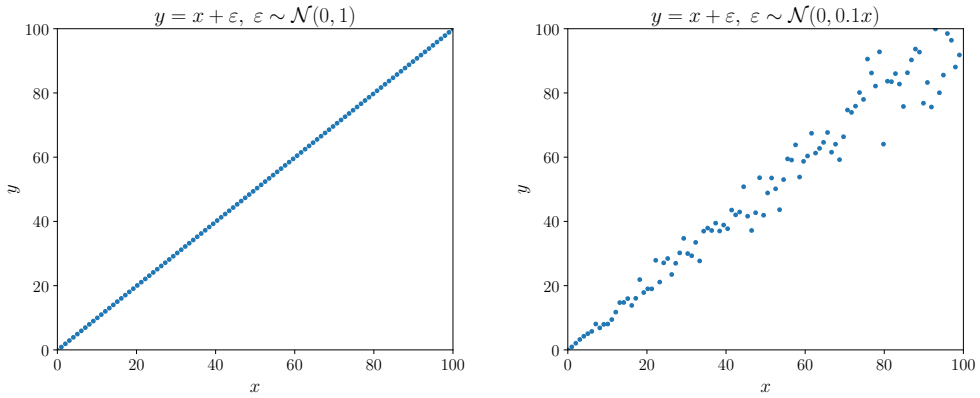
Figure 2.2 shows an illustration of the different components of the epistemic uncertainty and represents a hypothetical setting where we have induced a hypothesis $\hat{m}$ by training a machine learning model. The space $\mathcal{X} \times \mathcal{Y}$ represents the joint probability distribution of the data, and the perfect model for explaining the data is f. The hypothesis space $\mathcal{M}(\Theta)$ is a subset of the space representing the data, and the best hypothesis within the hypothesis space is m^*. We see that there is a discrepancy between the induced hypothesis $\hat{m}$ and the best hypothesis m^* within the hypothesis space. This discrepancy represents the approximation uncertainty. Furthermore, there is a discrepancy between the perfect model f and the hypotheses in the hypothesis space $\mathcal{M}$. This represents the model uncertainty.

The epistemic uncertainty can also be seen as an indicator of OOD test data. We have already seen that in such situations, we cannot extrapolate from the training to the test data, making it hard to reason about the ability of the algorithm to generalize to unseen data. Moreover, such situations are affiliated with great uncertainty. Let us consider a binary classification task, where the goal is to classify pictures of cats and dogs. The classifier is trained using pictures of cats and dogs only, and has thus not learned how to differentiate between other things. If we now test the algorithm using pictures of umbrellas, the algorithm should output a high level of uncertainty (if properly trained). On the other hand, if we augment the training data to include pictures of umbrellas, the algorithm learns to differentiate between cats, dogs and umbrellas, and the uncertainty is resolved. With careful inspection of the epistemic uncertainty, we can decide whether or not it is appropriate to augment the training data and re-train our model to cover a broader distribution of the data.

The dependency between the instance space $\mathcal{X}$ and the target space $\mathcal{Y}$ is not necessarily deterministic. That is, an element $X \in \mathcal{X}$ may give rise to a different set of elements $\boldsymbol{y} \in \mathcal{Y}$. This is related to the inherent randomness of the data-generating process we are modelling or random measurement errors when collecting the data. Thus, the corresponding uncertainty is attributed to the aleatoric uncertainty component, as it consists of the uncertainty related to the data-generating process.

The aleatoric uncertainty component is further decomposed into two different types, namely *homoscedastic* and *heteroscedastic* aleatoric uncertainty. The former represents situations where the aleatoric uncertainty is constant, and the latter varies among samples in the dataset. Modelling the aleatoric uncertainty in a homoscedastic manner is rather restrictive, because in most cases the uncertainty is expected to vary depending on e.g. the location of the measurement. In such situations, it is desirable to model the aleatoric uncertainty in a heteroscedastic manner to avoid loosing valuable information.

In Figure 2.3 we see an example where we have a linear relationship between the response variable y and the single explanatory variable x. In the homoscedastic case (left panel), we see that the additive noise is independent of the value of the regressor. In the heteroscedastic case (right panel), we see that the additive noise depends linearly on the value of the regressor, such that the level of noise increase with the value of x.



(a) Homoscedastic noise (b) Heteroscedastic noise

Figure 2.3: (a) Homoscedastic and (b) heteroscedastic aleatoric uncertainty for a linear relationship between the response y and regressor x. The data-generating process is marked in the title of each plot.

In machine learning applications where the models provide uncertainty measures, the different sources of uncertainty are typically not distinguished [27]. In fact, in some cases, the distinction is not necessary. If a machine learning model is used in decision support, the source of uncertainty is typically not relevant. If the level of uncertainty is above some threshold, the instance is forwarded to a human for further inspection, making the origin of the uncertainty irrelevant. In other situations, it might be helpful to investigate whether the elevated uncertainty levels arise from measurement errors, a misspecified model or simply by having little data. To reason about such, it is necessary to decompose the total predictive uncertainty into its aleatoric and epistemic components.

Referring back to Section 1.3, the irreducible aleatoric uncertainty represents the inherent randomness of the process being modelled, while the epistemic uncertainty is reducible and represents the lack of knowledge of the process. Suppose we observe that the predictive uncertainty decrease when we observe more data, i.e. obtain more knowledge of the process. In that case, there is evidence that we do not fully understand the data-generating process. This can be used to decide whether or not it is appropriate to gather more data.
If we can fit a model that has perfect knowledge of the data-generating process, the epistemic uncertainty vanishes. The total predictive uncertainty will, in this case, only consist of the irreducible, aleatoric uncertainty. Looking at the predictive uncertainty in this setting allows us to investigate the inherent randomness of the process. Having a model that can perfectly explain the data-generating process is not really possible. To be able to reason about the inherent randomness of the process, it is thus necessary to decompose the predictive uncertainty and extract the aleatoric and epistemic uncertainty components. This decomposition allows the practitioner to become aware of the process's stochastic behaviour and assert whether one should specify a different model or gather more data.

We have seen that the epistemic uncertainty covers the uncertainty related to the model and its parameters and is thus an attribute of the model itself. Moreover, we can use epistemic uncertainty to detect out-of-distribution test data. On the other hand, aleatoric uncertainty refers to the uncertainty in the data and is thus independent of the model.

Having a model that can reason about the uncertainty is valuable, even more so if the model can decompose the uncertainty into its different components.

2.2 Linear Regression

This section presents the bare minimum of theory regarding model assumptions in linear regression models and how one can use diagnostic plots to assert whether the assumptions are met. The reader is referred to [13] for a more thorough description.

Linear regression is arguably the most simple method for doing supervised learning, and is a widely used statistical method. The goal of linear regression is to fit a linear model that explains the relationship between a set of explanatory variables $\boldsymbol{x}$ and a target variable $\boldsymbol{y}$. The most common definition of the model is given by

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i, \ i = 1, \ldots, n \\ \varepsilon_i &\sim \mathcal{N}(0, \sigma^2), \end{aligned} \tag{2.3}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p) \in \mathbb{R}^{(p+1)}$ is the parameters of the model, $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \mathbb{R}^p$ the set of explanatory variables for the ith sample and n the number of samples. The last term in (2.3) is an additive error term, and p is the number of explanatory variables.

Using vector notation, we can rewrite the model as

$$\begin{aligned} \boldsymbol{y} &= X\boldsymbol{\beta} + \boldsymbol{\varepsilon} \in \mathbb{R}^n, \\ \boldsymbol{\varepsilon} &\sim \mathcal{N}_n(\mathbf{0}, \sigma^2 I) \in \mathbb{R}^n, \end{aligned}$$

where $X \in \mathbb{R}^{n \times (p+1)}$ is the so-called design matrix with rows corresponding to the samples in the dataset, and columns corresponding to the explanatory variables (including the intercept). The model definition leads to the following distribution for the target variable

$$\boldsymbol{y} \sim \mathcal{N}_n(X\boldsymbol{\beta}, \sigma^2 I).$$

The model is fit to the data through estimation of the model parameters such that the sum of the squared deviations between the target and the prediction is minimized. Two common ways of obtaining the estimate of the model parameters are Ordinary Least Squares (OLS) and Maximum Likelihood Estimation (MLE). In fact, in the case of a normally distributed error term, these approaches are equivalent, resulting in identical estimates [13]. In either case, the parameters are estimated as follows.

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}.$$

The linear regression model is concerned with a set of model assumptions, and one must be careful to assert whether or not the assumptions are met when using the model to explain the data at hand.

- Linearity assumption: The relationship between the explanatory variables and the target is linear in the parameters of the model
- Normality assumption: Normally distributed error term, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)\ \forall i$
- Homoscedasticity assumption: The error term is distributed with constant variance σ^2
- Multicollinearity assumption: The explanatory variables are not highly correlated

The distributional assumption on the error term allows to draw probabilistic statements about the model fit, e.g. how uncertain the model is when making predictions on unseen data.

After fitting a linear model to the data at hand, one can use diagnostic plots to look into the model assumptions. Three commonly used diagnostic plots are

- Residual plot
- Scale-location plot
- Normal Q-Q plot

The residuals are particularly useful for assessing the model assumptions, as they contain the variation in the data that is not explained by the explanatory variables. The residuals in a linear model are estimates of the unobserved error $\boldsymbol{\varepsilon}$ [13], and are defined as the difference between the true and predicted values for the target variable.

$$\begin{aligned} \hat{\boldsymbol{\varepsilon}} &= \boldsymbol{y} - \hat{\boldsymbol{y}} = (1 - H)\boldsymbol{y}, \\ \hat{\boldsymbol{y}} &= X\hat{\boldsymbol{\beta}} = H\boldsymbol{y}, \end{aligned} \tag{2.4}$$

where $H = (X^T X)^{-1} X^T$ is the so-called hat- or prediction matrix.

To assert whether the linearity assumption is met, one can look at the residual plot [29], which is obtained by plotting the residuals $\hat{\boldsymbol{\varepsilon}}$ against the fitted values $\hat{\boldsymbol{y}}$. Suppose there is a non-linear trend in the residual plot. In that case, there is evidence of the linear model not being capable of capturing the non-linear effects between the explanatory variables and the target. In this case, a non-linear model may be more suited for explaining the variation in the data at hand.

It is not adequate to use the residual plot for asserting the homoscedastic error assumption. This is becasuse the residuals are inherently heteroscedastic [13, p. 183]. One can show this by calculating the covariance of the residuals in (2.4).

$$\begin{aligned}\text{Cov}[\hat{\boldsymbol{\varepsilon}}] &= \text{Cov}[(1-H)\boldsymbol{y}] \\ &= (1-H)\text{Cov}[\boldsymbol{y}](1-H)^T \\ &= (1-H)\sigma^2 I(1-H)^T \\ &= \sigma^2(1-H),\end{aligned}$$

as the matrix $(1-H)$ is symmetric and idempotent [13, p. 122]. For the variance of the residuals, we then have

$$\begin{aligned}\text{Var}[\hat{\varepsilon}_i] &= \sigma^2(1-H_{i,i}) \\ &= \sigma^2(1-[X(X^TX)^{-1}X^T]_{i,i}),\end{aligned} \tag{2.5}$$

where $H_{i,i}$ denotes the ith diagonal element of the matrix H.

From (2.5) it is clear that the residuals are potentially varying for different samples i, and one can thus not use the raw residuals for checking the homoscedasticity assumption.

To fix the problem of heteroscedastic residuals, one must scale the residuals to a common variance, thus obtaining the standardized residuals.

$$r_i = \frac{\hat{\varepsilon}_i}{\hat{\sigma}\sqrt{1-H_{i,i}}},\ i = 1,\ldots,n,$$

where $\hat{\sigma}$ is the estimated standard deviation of the raw residuals. For the above standardization to hold, it is important to note that the expected value of the residuals is zero.

$$\begin{aligned}\mathbb{E}[\hat{\boldsymbol{\varepsilon}}] &= \mathbb{E}[\boldsymbol{y}] - X(X^TX)^{-1}X^T\mathbb{E}[\boldsymbol{y}] \\ &= X\boldsymbol{\beta} - X(X^TX)^{-1}X^TX\boldsymbol{\beta} \\ &= X\boldsymbol{\beta} - X\boldsymbol{\beta} = \mathbf{0}.\end{aligned}$$

Provided that the model assumptions are correct, i.e. that the error variance is homoscedastic, the standardized residuals exhibits a constant spread. This is examined using the scale-location plot, where the square root of the standardized residuals are plotted against the fitted values.

To assert whether the normality assumption is met, one can look at the Normal Q-Q plot, which is a plot of the observed quantiles of the standardized residuals against the theoretical quantiles of the standard normal distribution. If the standardized residuals follow a diagonal line with a unit slope, the observed quantiles for the standardized residuals equals the theoretical quantiles of the standard normal distribution. One can then conclude that the standardized residuals are distributed as a standard normal distribution.

2.3 Neural Networks

Neural networks are a broad class of machine learning models whose goal is to map a functional relationship between a set of input features to an output.
There exist a great number of different neural network models, and this section is devoted to the general class of Feedforward Neural Networks (FNN), the quintessential deep learning model [18].

A FNN model can be used to approximate a function f depending on a set of parameters $\boldsymbol{\theta}$, and the network is trained to learn the values of the parameters that best fit the function being approximated. The parameters of the model are typically referred to as *weights* when describing neural networks. The terms weights and parameters will be used interchangeably throughout this section.

Feedforward neural networks are constructed in a layer-wise manner, where each *layer* consists of computational units typically referred to as *neurons*. The output of a single neuron is simply a weighted sum of the input features, plus a bias term. The weights are given by the parameters of the neuron. This will, however, only allow the models to approximate linear functions between the input and output. To allow for non-linear functional relationships, the output is coupled with a non-linear *activation function*. A commonly used activation function is the ReLU activation, an abbreviation for Rectified Linear Unit, a piecewise linear function. The activation function is expressed as follows.

$$\sigma_{\text{ReLU}}(z) = \max\{0, z\}, \tag{2.6}$$

where z is the output of a neuron, or the so-called pre-activation value. Neural networks that are equipped with ReLU activation in their intermediate layers are referred to as rectified networks.

A computational graph of a single neuron with input vector $\boldsymbol{x} = (x_i, x_2)$, bias b and activation function σ is shown in Figure 2.4.

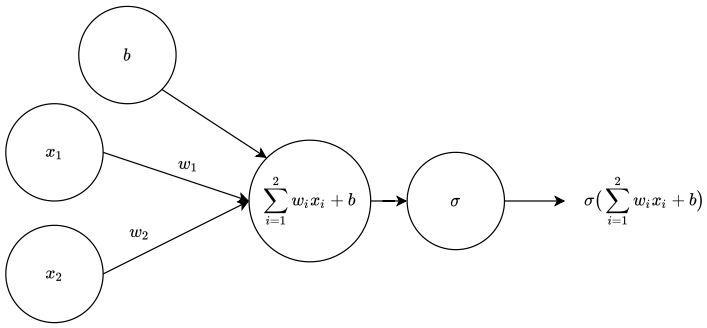


Figure 2.4: Computational graph of a single neuron in a neural network, with input $\boldsymbol{x} = (x_1, x_2)$, weights $\boldsymbol{w} = (w_1, w_2)$ and bias b. The activation function is σ.

As the name suggests, feedforward neural networks are built in a feedforward manner, where the output of a layer is fed as input to the subsequent layer. It is important to note that the output of a single neuron in a layer is input to all the neurons in the subsequent layer. Consequently, the layer is said to be *fully connected*. The final output of a neural network with multiple layers is thus a composition of multiple linear functions, typically coupled with non-linear activation functions to allow for more complex functional relationships.

The intermediate layers of a FNN, i.e. the layers between the input and output layers, are referred to as the *hidden layers* of the model. The term resides from the

fact that the training data does not specify the output of these intermediate layers during training, unlike the input and output layers[3] [18].

A FNN model is typically viewed as a directed acyclic graph (DAG), describing how the functions corresponding to the different layers are composed to obtain the output from the input. A simple single-layer FNN can be seen in Figure 2.5, where an input vector $\boldsymbol{x} = (x_1, x_2)$ is passed through a hidden layer $\boldsymbol{h} = (h_1, h_2, h_3)$ with three neurons to obtain a scalar output $\hat{y}$. The network parameters, i.e. the weights and biases for the linear combinations of the neurons, are contained in the weight matrices W_1, W_2.

Following the DAG in a feedforward manner we can express the output as

$$\begin{aligned}
&\hat{y} = W_2^T \boldsymbol{h} + \boldsymbol{b_2} = W_2^T(\sigma(W_1^T \boldsymbol{x} + \boldsymbol{b_1})) + \boldsymbol{b_2} \in \mathbb{R}, \\
&W_1 \in \mathbb{R}^{2\times 3},\ \boldsymbol{x} \in \mathbb{R}^2,\ \boldsymbol{b_1} \in \mathbb{R}^3, \\
&W_2 \in \mathbb{R}^{3\times 1},\ \boldsymbol{h} \in \mathbb{R}^3,\ \boldsymbol{b_2} \in \mathbb{R},
\end{aligned}$$

where σ is the activation function for the hidden layer, and $\boldsymbol{b_1}, \boldsymbol{b_2}$ is the bias to the hidden layer and the output, respectively.

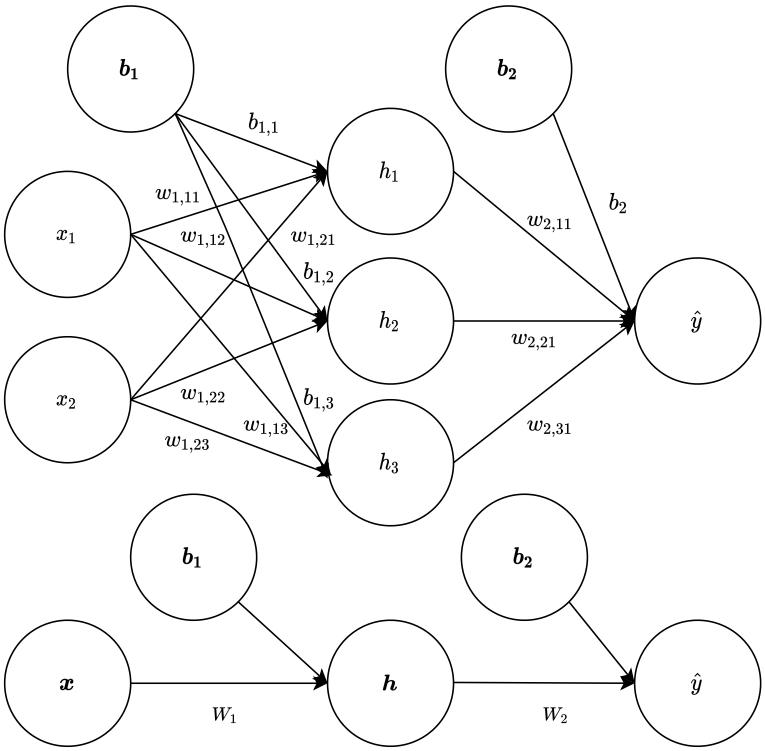


Figure 2.5: Computational graph of a Feedforward Neural Network with a single hidden layer $\boldsymbol{h} = (h_1, h_2, h_3)$ for instance $\boldsymbol{x} = (x_1, x_2)$. The scalar output of the FNN is $\hat{y}$, and the paramters are contained in the weight matrices W_1, W_2 and the bias vectors $\boldsymbol{b_1}, \boldsymbol{b_2}$.

[3] in the case of supervised learning

The scalar output $\hat{y}$ of a generic FNN with input $\boldsymbol{x}$ can be expressed as

$$\hat{y}(\boldsymbol{x}) = f^{(1)}(f^{(2)}(\dots f^{(L)}(\boldsymbol{x}))),$$

where L is the *depth* of the neural network, i.e. the number of hidden layers. The function $f^{(i)}$ denotes the function corresponding to layer i, including potential bias terms and activation functions. Note that each layer has a separate weight matrix W_i, containing all the parameters of the linear combinations given by the neurons in the layer.

The parameters of a neural network are learned by optimizing a suitable loss function, which is closely related to the activation function in the output layer. The type of activation function is furthermore related to the type of problem being solved. One can show that optimizing a specific loss function is equivalent to setting up a conditional distribution for the output and perform maximum likelihood estimation on the parameters. An example for a regression problem follows.

In a regression setting, where to goal is to output a scalar value, a linear activation is used in the output layer. The linear activation function is simply a unit transformation of its input.

$$\hat{y}(\boldsymbol{x}) = \sigma_{\text{Linear}}(W_L^T \boldsymbol{h}^{(L)} + \boldsymbol{b}^{(L)}) = W_L^T \boldsymbol{h}^{(L)} + \boldsymbol{b}^{(L)}, \tag{2.7}$$

where $\boldsymbol{h}^{(L)}(\boldsymbol{x}) = f^{(1)}(f^{(2)}(\dots f^{(L-1)}(\boldsymbol{x})))$ is the feature vector of layer $L-1$ for input $\boldsymbol{x}$. The bias vector for the final layer is $\boldsymbol{b}^{(L)}$, and W_L is the corresponding weight matrix.

We can formulate the regression problem as modelling the conditional mean of a normal distribution with mean $\hat{\boldsymbol{y}}$ and variance σ^2,

$$p(\boldsymbol{y}|X) = \mathcal{N}_n(\hat{\boldsymbol{y}}, \sigma^2),$$

with log-likelihood

$$\log p(\boldsymbol{y}|X) = -\frac{n}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

where n is the number of samples in the dataset. In the above formulation the conditional mean $\hat{\boldsymbol{y}}$ contains the predictions for all the samples in the dataset, and the samples are gathered in the design matrix $X \in \mathbb{R}^{n\times p}$ where p is the number of explanatory variables.

The Maximum Likelihood Estimate (MLE) of the parameters in the network is calculated as follows.

$$\begin{aligned}
\boldsymbol{\theta}^{\text{MLE}} &= \underset{\boldsymbol{\theta}}{\text{argmax}} -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\
&= \underset{\boldsymbol{\theta}}{\text{argmin}} \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\
&= \underset{\boldsymbol{\theta}}{\text{argmin}}\, MSE(\boldsymbol{y}, \hat{\boldsymbol{y}}),
\end{aligned}$$

where the parameter $\boldsymbol{\theta}$ consist of all the weights and biases for all the layers in the network.

Minimizing the Mean Squared Error (MSE) loss function[4] in a regression setting is thus equivalent to performing maximum likelihood estimation of the parameters of a neural network with a linear activation function in the output layer.

[4]with respect to the parameters of the neural network

When an input instance $\boldsymbol{x}$ with corresponding target y is passed through the network to produce an output $\hat{y}$, the information flows in a forward direction from input to output. The operation is called a *forward pass* through the network. During the training of the model, the forward pass continues until obtaining a scalar loss $\mathcal{L}(y, \hat{y})$. Then, the network parameters are updated in an iterative manner using gradient-based learning, minimizing the loss by utilizing the gradient of the loss with respect to the parameters. The gradients are calculated using the *backpropagation* algorithm, which is based on the chain rule of calculus. The updating is done using the gradient descent algorithm with a specified learning rate or one of its many varieties. A commonly used optimization algorithm for training neural networks is the Adam algorithm [31], a gradient descent like algorithm where the learning rate is adaptively updated during training. During an iteration of the gradient descent algorithm, the parameters $\boldsymbol{\theta}$ are updated as follows.

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \alpha \nabla_{\theta_i} \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}),$$

where α is the specified learning rate. Note that in the above gradient descent step, the gradient of the loss is computed over the entire training set. The computation quickly becomes expensive for complex models with a lot of parameters and large datasets, and does not scale well to such situations. In this situation, the parameters are updated once every pass through the entire training set, otherwise known as an *epoch*.

To be able to scale to larger models and datasets, it is common practice to use *mini-batch* optimization, where the parameters are updated multiple times during an epoch, on randomly sampled batches of the entire training set. It can be shown that mini-batch gradient descent obtains an unbiased estimate of the total gradient by computing the average gradient over the samples in the mini-batch [18]. The number of samples in a mini-batch is specified by the *batch size*. A complete pass over the training set, i.e. an epoch, consists of a number of *iterations* for updating the parameters.

For mini-batch gradient descent with a batch-size of b, the updating during a single iteration is done as follows

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \alpha \frac{1}{b} \sum_{j=1}^{b} \nabla_{\boldsymbol{\theta}_i} \mathcal{L}(y_j, \hat{y}_j), \ i = 1, \ldots, M, \tag{2.8}$$

where M is the number of iterations in an epoch, i.e the number of times the parameters are updated during a full pass over the training set.

The non-linearity posed by the activation functions in neural network models causes the most interesting loss functions to become non-convex, meaning that there is no global convergence guarantee of the solution for the parameters [18].

A common problem when training neural networks is vanishing gradients, where the gradient of the loss function with respect to the parameters in the network becomes close to zero. This is causing the updating of the parameters to stall, resulting in sub-optimal convergence of the optimization. A related problem occurs when the gradients take too large values, known as the exploding gradient problem. The problems are typically gathered under the collective term Exploding and Vanishing Gradient Problem (EVGP) [22].

Stochastic gradient descent, such as the above mini-batch optimization (2.8), applied to non-convex optimization objectives, is very sensitive to the initial values of the parameters, and it is thus important to initialize the parameters appropriately.

The authors in [24] propose a robust initialization method for the weights and biases in rectified neural networks, known as the He-initialization after one of the authors. With He-initialization, the weights in the neural network are initialized by drawing samples from a zero-mean normal distribution with standard deviation $\sigma = \sqrt{\frac{2}{n_l}}$ for the lth layer, where n_l is the number of inputs values to the activation. Since no ReLU activation is applied to the input signal, the weights in the first layer are initialized by drawing samples from a zero-mean Gaussian distribution with standard deviation $\sigma = \sqrt{\frac{1}{n_{in}}}$, where n_{in} is the number of explanatory variables. The biases are all initialized to zero. With this parameter initialization, a unit variance for the post-activation values is obtained, which resolves the problem of exploding and vanishing gradients [35]. Note that this approach only depends on the size of the network.

During training, the distribution of the input of each layer changes as the parameters are updated. This is causing the neural network architecture to become even more sensitive to the initialization of the parameters and slows down training by requiring lower learning rates [28]. The resulting phenomena is referred to as *internal covariate shift*. In [28] the authors propose a method to reduce the internal covariance shift and thus accelerate training and making the architecture more robust to parameter initialization. The method is known as *batch normalization* (BN). In batch normalization, the output of the layer is normalized for every mini-batch and scaled and shifted by two additional parameters introduced by the BN layer. The additional parameters are learned alongside the network parameters. The BN algorithm for a single mini-batch is summarized in Algorithm 1, where ε is a small constant added to the mini-batch variance to obtain numerical stability. For more technicalities about the method, the reader is referred to [28].

Algorithm 1: Batch normalization for a single mini-batch $\mathcal{B}$. Adapted from [28].

Inputs : Mini-batch $\mathcal{B} = \{x_i\}_{i=1}^m$
Learnable parameters γ, β
Output: Transformed output $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}_{i=1}^m$

Compute mini-batch statistics
$\mu_\mathcal{B} = \frac{1}{m}\sum_{i=1}^m x_i$
$\sigma_\mathcal{B}^2 = \frac{1}{m}\sum_{i=1}^m (x_i - \mu_\mathcal{B})^2$
Normalize the input, and scale the resulting normalized values
for $i \leftarrow 1$ **to** m **do**
  $z_i = \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \varepsilon}}$
  $y_i = \gamma \cdot z_i + \beta$

Machine learning models based on deep neural networks have an incredibly high capacity and are thus prone to overfitting. A central challenge in machine learning is to generalize to unseen data, i.e. data not seen during training (see 2.1). If a model is overfitting the training data, the training error becomes very low, while the generalization abilities become very poor. Thus, it is necessary to include some regularization when training the model, which controls the capacity and prevents overfitting. Two widely used regularization techniques are L^2 regularization and dropout.

With L^2 regularization, a regularization term is added to the loss function. L^2 regularization is equivalent to ridge regression seen in statistical learning, and the idea is to shrink the parameters by imposing a penalty on their magnitude into the loss function [23]. The penalty parameter in L^2 regularization applied to neural network models is referred to as the *weight-decay* parameter. We express the loss function for training a neural network of depth L with L^2 regularization as follows.

$$J(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda \boldsymbol{w}^T \boldsymbol{w},$$

where λ is the weight-decay parameter, $\boldsymbol{w} = (W_1, W_2, \ldots, W_L)$ denotes the vector of weight matrices in the neural network, $\hat{\boldsymbol{y}}$ is the predicted target variable for an instance with corresponding groundt-truth target $\boldsymbol{y}$ and $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is the loss function. We see that the magnitude of the weights are expressed in terms of their L^2-norm, hence the name of the regularization technique.

Dropout regularization can be applied to a broad range of neural network models and is computationally inexpensive and powerful [18]. However, the technique is restricted to neural networks.

The idea behind dropout is to randomly drop neurons from the neural network during training [52], thus reducing the representational capacity. By dropping a neuron from the network, all weights and biases corresponding to the dropped neuron are set to zero and will not contribute to the overall output. Training a neural network with dropout introduce an additional hyper-parameter to be set before training, namely the *dropout probability*. When a neural network is trained with dropout, a single forward pass through the network is equivalent to sampling a thinned network from a population of all thinned networks that can be formed by removing neurons from the non-output layers of the full network [18], and passing the input instance through the sampled network. The rationale behind the technique is to pass each instance through the network multiple times, thus obtaining a distribution of predictions resulting from multiple thinned networks, and take the mean value of this distribution as the prediction to obtain a better prediction for each instance. During test time, i.e. when using the trained model to make predictions, the effect of averaging the predictions from the thinned networks is approximated by using the entire network with downscaled weights [52]. Doing so supersedes the need of doing multiple forward passes through the network for each input instance, thus reducing the computational complexity of the method. The downscaling of the weights are performed simply by using the dropout probability of each layer.

It should be clear that setting up and training a feedforward neural network leaves the practitioner with numerous choices. The choices relevant to the above description of neural networks are summarized below. Before training the model, the practitioner will have to specify some particular parameters that will not be updated during training, namely the hyper-parameters.

First of all, the practitioner needs to find an appropriate loss function and output activation for the problem at hand. Next, the user has to specify the network *architecture*, referring to the model structure in terms of the number of hidden layers and neurons in each layer. The architecture is directly related to the model's representational capacity, and the size and structure of the dataset need to be taken into account at this point. A rule of thumb is that a larger dataset allows for more complex models with more computational units, i.e. more hidden layers and neurons, than a smaller dataset. To allow for non-linear dependencies within the hidden layers, the user will have to specify suitable activation functions between the

layers. Further on, the model is typically trained using gradient-based optimization, and the user needs to specify a suitable optimization method for doing so. An important hyper-parameter introduced by the optimizer is the learning rate. Setting the learning rate too high may result in unstable training or fast convergence to a poor solution. On the other side, using a too low learning rate may result in slow convergence or the solution being stuck in a local minimum [8]. It is thus important to find the optimal learning rate to train the model correctly. However, when using optimizers with adaptive learning rates, like the already mentioned Adam algorithm, setting the learning rate to its optimal value becomes less critical.

Other important hyper-parameters that need to be set are the number of epochs and the batch size in mini-batch optimization. These will decide the number of iterations used for updating the model's parameters and are related to the model's capacity because a model trained on fewer epochs is more prone to under-fitting than a model being trained for more epochs. Another way to effectively control the model's capacity is by using regularization, which typically requires additional hyper-parameters to be set. In the case of L^2 and dropout regularization, the additional hyper-parameters are the weight-decay parameter and the dropout probability.

The values of the hyper-parameters are typically found in an empirical way using trial and error. However, there exist numerous techniques for finding the optimal values automatically, e.g. using a grid-search over the space of hyper-parameters [18]. The idea behind grid-search is to train a model for every specification of the hyper-parameters in the cartesian product of the set of possible values and select the set of values for the hyper-parameters that yield the best predictive performance a held-out validation set. When the optimal configuration of hyper-parameters is found, the model is re-trained using the entire training set (including the validation set). A grid-search of the space of hyper-parameters is thus concerned with training multiple models, which quickly becomes computationally expensive for large models with many hyper-parameters.

2.3.1 Preprocessing and Feature Engineering

Real-world datasets are susceptible to noisy measurements and missing data due to measurement error. To deal with this, it is necessary to preprocess the data before feeding it into the neural network, making it suitable for training the model.

Neural networks cannot deal with missing values, so they need to be appropriately taken care of. There are numerous methods to deal with missing values, and the most simple is to ignore the missing values simply by removing the instances that contain missing values for one or more features. A more tedious approach is to manually insert the missing values by using domain knowledge and extrapolation. However, for large datasets, this approach becomes very impractical. Another approach is to use a global constant to insert into the missing values. This can e.g. be a constant that represents an unknown value that the model can process, hoping that the model can interpret the values as missing. In the case of many missing values, this approach may cause the model to misinterpret the unknown constant as an interesting property of the data, which may result in poor performance. Dealing with missing values by insertion is referred to as *missing value imputation*. A commonly used variety of this constant imputation is to use an adequately chosen statistic based on known values of the missing feature in the dataset. This can e.g. be the mean or median value of the missing feature over the dataset. There exist other, more sophisticated methods for imputing missing values. An example is training a model where the missing value is the response variable, and the other variables for the instance are the explanatory variables.

Another critical preprocessing step in deep learning is to normalize the input features to a common scale [21]. Normalizing the data aims at giving all features an equal weight, such that the scale, i.e. the unit of measurement of each feature, is irrelevant to the modelling. This is typically done in the case of neural networks, as it speeds up convergence of the optimization [21, 40]. Normalizing the data is performed by replacing each feature with its corresponding z-score, having zero mean and unit variance. The z-score of a feature is calculated as follows.

$$z_i = \frac{x_i - \mu}{\sigma},\ i = 1, \ldots, N, \tag{2.9}$$

where $\boldsymbol{x} = \{x_i\}_{i=1}^{N}$ is the feature that is normalized, μ the mean value and σ the standard deviation of the feature over the dataset with N samples.

It is important to note that a preprocessing scheme is specific to the data and problem at hand. Having a preprocessing scheme that is perfectly suitable for a given setting does not necessarily mean that it is suitable in another setting.

Another potentially important step of the modelling pipeline is to create new features based on the features already present in the dataset. This technique is referred to as *feature engineering*. The idea is to expand the feature set by including new features relevant to the modelling process, potentially making the model perform better. Feature engineering is typically based on domain knowledge, where a specialist carefully examines the original features and decides whether e.g. the difference or a fraction of two features is relevant for the modelling. The process is labour-intensive but may yield better predictive performance than what the original feature set provides.

The field of deep learning and neural networks encompasses much more than what is described in this section. For a more thorough and complete description, the reader is referred to the excellent book by Goodfellow, Bengio and Courville [18].

In general, and specifically in the context of this thesis, it is important to note that the trained parameters of neural network models are point estimates. Moreover, there is typically no distributional assumption on the error term like we see in more traditional statistical methods, e.g. linear regression (see Section 2.2). Consequently, the models are deterministic. If the same input instance is forwarded through the network multiple times, the output will be constant. These models are thus incapable of assessing their predictive uncertainty, and they tend to make overly confident predictions [7].

Reasoning about the predictive uncertainty of a neural network model requires equipping the network parameters with probability distributions, and the calculations are performed using Bayesian inference.

2.4 Bayesian Inference

Bayesian inference is the process of inductive learning using Bayes rule [26], and provides the mathematical framework for updating a prior expectation about a hypothesis given observed evidence. The formula for Bayes rule is defined as follows

$$p(\mathcal{H}|E) = \frac{p(\mathcal{H})p(E|\mathcal{H})}{p(E)}, \tag{2.10}$$

where $\mathcal{H}$ is the hypothesis and E the observed evidence.

Bayes rule (2.10) contains the following components, and their respective names will be used throughout this section.

- **Prior distribution** $p(\mathcal{H})$
  Capturing the prior expectation of the hypothesis before observing any evidence
- **Posterior distribution** $p(\mathcal{H}|E)$
  The probability of the hypothesis given the observed evidence
- **Likelihood** $p(E|\mathcal{H})$
  The probability of observing the evidence, given a fixed hypothesis
- **Marginal likelihood** $p(E)$
  The probability of observing the evidence, regardless of the hypothesis

Bayesian inference applied to modelling is often concerned with inferring the posterior distribution of a parameter θ given observed data X, where the distribution of X depends on the parameter θ. Hence the following formulation of Bayes rule:

$$p(\theta|X) = \frac{p(\theta)p(X|\theta)}{p(X)} = \frac{p(\theta)p(X|\theta)}{\int_{\Theta^*} p(X|\theta^*)p(\theta^*)d\theta^*}. \tag{2.11}$$

In the denominator of the last equality in equation (2.11), observe that the marginalization is done over the set of all possible parameters $\theta^* \in \Theta^*$. The marginal likelihood $p(X)$ does not depend on the parameter θ, and one can write

$$p(\theta|X) \propto p(\theta)p(X|\theta).$$

The posterior is thus proportional to the prior times the likelihood. This is particularly useful when the distribution of the data-generating process is known, and the prior and posterior are conjugate distributions, where the posterior follows the same parametric form as the prior.

The Bayesian *credible interval* represents the probability of the parameter being contained in the interval. The credible interval is defined as follows.

$$\int_{\mathcal{C}} p(\theta^*|X)d\theta^* = 1 - \alpha,$$

where $\mathcal{C}$ is the credible interval, and $1 - \alpha$ is the probability of the parameter being contained in $\mathcal{C}$. The above definition is however ambigous, and one need to impose further restrictions to obtain a single credible interval. An example of such is the Highest Posterior Density (HPD) interpretation of the credible interval [4].

There is a clear difference in interpretation between the Bayesian credible interval and its frequentist cousin, the confidence interval. In the frequentist

confidence interval, the parameter is fixed, and the interval is treated as random. The corresponding confidence level, given by $1 - \alpha$, represents the long-term frequency of the interval containing the parameter's actual value. On the other hand, in the Bayesian credible interval, the parameter is treated as a random variable while the interval is fixed. This allows making direct probabilistic interpretations of the credible interval.

In large-sample situations, Berger argues that the Bayesian approach is almost always equivalent to the frequentist [4, p. 125]. However, their differences in interpretation remain. It is possible to use this analogy to evaluate credible intervals in terms of the confidence interval's frequentist coverage probability. This means that one can expect the true value of the parameter being estimated to fall within the $(1 - \alpha)$ credible interval a fraction of $(1 - \alpha)$ times in the long run. This is often used to evaluate the robustness of Bayesian approaches, e.g. in terms of the choice of prior distributions.

A particularly useful tool in the field of Bayesian inference is the *posterior predictive distribution*, which describes the posterior probability of new, unobserved data conditional on the observed data. The posterior predictive distribution is calculated by marginalizing the distribution of the new data conditional on a fixed parameter, over the posterior distribution, thus accounting for the uncertainty related to the parameter.

$$\begin{aligned} p(\tilde{x}|X) &= \int_{\Theta^*} p(\tilde{x}|\theta^*, X)p(\theta^*|X)d\theta^* \\ &= \int_{\Theta^*} p(\tilde{x}|\theta^*)p(\theta^*|X)d\theta^*, \end{aligned} \tag{2.12}$$

where $\tilde{x}$ is the new, unobserved data of interest. Note that the last equality in (2.12) assumes that the new observation $\tilde{x}$ is conditionally independent to the already observed data X, given the parameter θ.

Going back to equation (2.11), we see that the marginalization in the denominator is done over the set of all possible values for the parameter θ. The integral can be evaluated analytically for trivial models, but the calculations quickly become intractable when the number of parameters in the model increase. The integration is typically over a combinatorically large space for complex models, and it is not possible to get a closed-form expression for the posterior distribution. For such models, approximation techniques are needed. The following section is devoted to describing a widely used technique for doing approximate inference, namely variational inference.

2.4.1 Variational Inference

The application of complex Bayesian models to practical problems involves calculating the typically intractable posterior distribution $p(\theta|X)$ with no closed-form solution. To deal with the intractability, the practitioner needs to resort to approximate inference.

For decades, the dominant method for doing approximate inference has been the Markov Chain Monte Carlo method (MCMC). In MCMC, an ergodic Markov chain is constructed, whose stationary distribution is the posterior distribution of interest. The posterior distribution is then approximated using an empirical estimate constructed from samples of the Markov chain [6]. Although MCMC is widely used

in industry and academia, it is computationally expensive and hard to scale to complex models and large datasets.

Variational inference (VI) provides a good alternative to approximate the posterior distribution[5]. Rather than using sampling to approximate the posterior, VI aims to obtain an approximate posterior using optimization. The goal is to approximate the posterior distribution $p(\theta|X)$ with a variational distribution $q_\rho(\theta)$ parametrized by ρ.

$$p(\theta|X) \approx q_\rho(\theta). \tag{2.13}$$

A simple schematic can be seen in Figure 2.6, where the true posterior (blue) is approximated by the variational distribution (red).

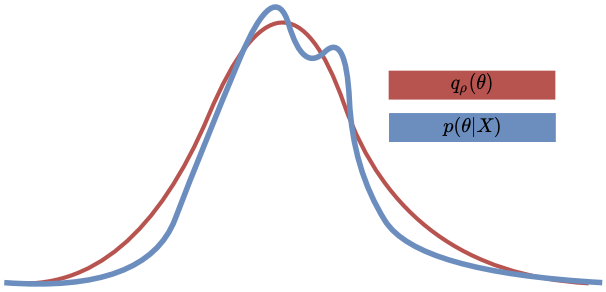


Figure 2.6: The variational distribution $q_\rho(\theta)$ in red approximating the true posterior $p(\theta|X)$ in blue.

.

The key idea of VI is to find a variational distribution (2.13) belonging to a family $\mathcal{Q}$ of approximate distributions over the parameter of interest, parametrized by a set of variational parameters ρ. The goal is then to find the optimal set of variational parameters such that the variational distribution approximates the true posterior well. The family $\mathcal{Q}$ of possible variational distributions is chosen by the practitioner. The idea is to restrict the complexity of the variational distribution to reduce the computational complexity related to the optimization while still approximating the true distribution well. A widely used variational family is chosen using the Mean-Field approximation, and the corresponding VI is often referred to as Mean-field variational inference. Without going much further into mean-field theory, the approximation assumes that the variational distribution can be factorized by assuming that the parameters of interest $\theta \in \Theta$ are mutually independent. A generic member of the variational family then takes the following form

$$q_\rho(\theta|\rho) = \prod_{i=1}^{m} q(\theta_i|\rho_i).$$

We want the variational distribution to be as close as possible to the true posterior, and this is achieved by minimizing the Kullback-Leibler (KL) divergence between the two distributions. By minimizing the KL divergence between the variational and the true posterior distribution, we minimize the information lost by approximating the true posterior with the variational distribution [5].

The optimization problem posed by VI applied to Bayesian modelling takes the following form

$$q_\rho(\theta)^* = \underset{q_\rho \in \mathcal{Q}}{\operatorname{argmin}}\, D_{\mathrm{KL}}[q_\rho(\theta)||p(\theta|X)], \tag{2.14}$$

[5]computing an approximate posterior is only a single instance of the general problem solved by VI, namely approximation of intractable integrals

where the KL divergence between two generic distributions $q(x)$ and $p(x)$ is defined as follows.

$$D_{\mathrm{KL}}[q(x)||p(x)] = \int_{\mathbb{R}} q(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]. \tag{2.15}$$

Using Bayes rule for the posterior distribution of the parameter we can rewrite the minimization objective in (2.14).

$$\begin{aligned}
D_{\mathrm{KL}}[q_\rho(\theta)||p(\theta|X)] &= \int_{\mathbb{R}} q_\rho(\theta) \log \frac{q_\rho(\theta)}{p(\theta|X)} d\theta \\
&= \int_{\mathbb{R}} q_\rho(\theta) \log \frac{q_\rho(\theta) p(X)}{p(\theta) p(X|\theta)} d\theta \\
&= \int_{\mathbb{R}} q_\rho(\theta) \log \frac{q_\rho(\theta)}{p(\theta)} d\theta \\
&\quad - \int_{\mathbb{R}} q_\rho(\theta) \log p(X|\theta) d\theta \\
&\quad + \log p(X) \int_{\mathbb{R}} q_\rho(\theta) d\theta \\
&= D_{\mathrm{KL}}[q_\rho(\theta)||p(\theta)] - \mathbb{E}_{q_\rho(\theta)}[\log p(X|\theta)] + \log P(X) \\
&= \log p(X) - \mathcal{L}(q_\rho(\theta)),
\end{aligned} \tag{2.16}$$

where $\mathcal{L}(q_\rho(\theta))$ is the evidence lower bound (ELBO), defined as

$$\mathcal{L}(q_\rho(\theta)) = \mathbb{E}_{q_\rho(\theta)}[\log p(X|\theta)] - D_{\mathrm{KL}}[q_\rho(\theta)||p(\theta)]. \tag{2.17}$$

The KL divergence in (2.16) cannot be minimized directly due to its dependency on the log evidence, which is, due to its intractability, the reason to use approximate inference in the first place [6]. On the other hand we see that the log evidence is constant with respect to the variational parameters ρ that determines the variational distribution. Minimizing (2.16) with respect to ρ is thus equivalent to maximizing the ELBO in (2.17) with respect to ρ, and we end up with the following optimization problem for finding the variational distribution.

$$q_\rho(\theta)^* = \underset{q_\rho \in \mathcal{Q}}{\operatorname{argmax}}\, \mathcal{L}(q_\rho(\theta)).$$

Note that the ELBO (2.17) only depends on the expected log-likelihood given by the data and the KL divergence between the variational and the prior distribution of the parameter θ. Thus, to minimize the KL divergence between the variational distribution and the true posterior, we only need the log-likelihood and the prior.

In the next section, Bayesian inference and deep learning are united by describing Bayesian neural networks. For a more detailed review of variational inference, the reader is referred to [6].

2.5 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) are similar to the deterministic feedforward neural networks described in Section 2.3. The difference is that the parameters of a BNN are represented by probability distribution rather than scalar values.

Introducing probability distributions to the parameters of a neural network allows capturing the uncertainty in the estimated parameters, thus allowing us to reason about the predictive uncertainty of the model.

Setting up a BNN requires the specification of a prior distribution of the parameters of the network, denoted as $p(\boldsymbol{w})$. The prior distribution is then updated using the likelihood of the data, denoted as $p(\mathcal{D}|\boldsymbol{w})$, as well as the evidence $p(\mathcal{D})$, to obtain the posterior distribution $p(\boldsymbol{w}|\mathcal{D})$ through the use of Bayesian inference. Every parameter in a BNN has a separate distribution to reflect the different contributions to the output.

A computational graph of a single-layer BNN is shown in Figure 2.7. The (posterior) distribution of the weights are shown as the red curves on the graph's edges. Note that the figure assumes no bias.

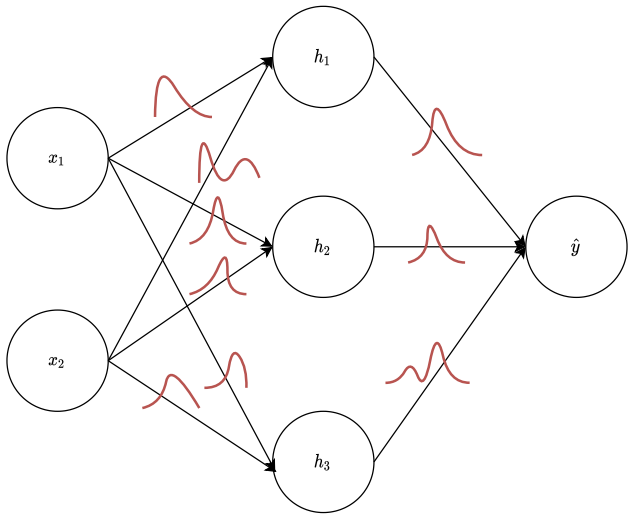


Figure 2.7: Computational graph of a Bayesian neural network with input $\boldsymbol{x} = (x_1, x_2)$, hidden layer $\boldsymbol{h} = (h_1, h_2, h_3)$ and output $\hat{y}$, The red curves on the edges of the graph represent the posterior distributions of the parameters in the network.

If we consider a regression task we can formulate a mathematical model for the target variable as follows.

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{z} + \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \sim \mathcal{N}_n(0, \sigma^2 I), \\ \boldsymbol{z} &= f(X, \boldsymbol{w}), \\ \boldsymbol{w} &\sim p(\boldsymbol{w}) = \prod_{i=1}^{K} \mathcal{N}(\mu_i, \sigma_i^2), \end{aligned} \tag{2.18}$$

where $\boldsymbol{y}$ is a measurement of the target variable being modelled by a Bayesian neural network $f(\cdot, \boldsymbol{w})$ with parameters $\boldsymbol{w}$ for all instances X in the dataset with size n. The number of parameters in the model is K, and the prior distribution for the parameters is a mean-field normal distribution with mean μ_i and variance σ_i^2 for $i = 1, \ldots, K$. The measurement $\boldsymbol{y}$ is subject to additive, homoscedastic noise $\boldsymbol{\varepsilon}$.

The model formulation in (2.18) allows setting up a conditional generative model for the measurement $\boldsymbol{y}$, given the explanatory variables X and the posterior distribution for the parameters $\boldsymbol{w}$.

$$\boldsymbol{y}|X, \boldsymbol{w} \sim \mathcal{N}_n(f(X, \boldsymbol{w}), \sigma^2 I). \tag{2.19}$$

When using a trained BNN to make predictions on an input instance, the input is fed through the network in a feedforward manner, just as for a deterministic FNN. However, the parameters are now being sampled from the posterior distributions of the parameters in the network. The prediction $\hat{y}$ for an input instance $\boldsymbol{x}$ using the above generative model (2.19) is obtained as follows.

$$\begin{aligned} \hat{y} &= f(\boldsymbol{x}, \boldsymbol{w}|\mathcal{D}) + \sigma^2, \\ \boldsymbol{w}|\mathcal{D} &\sim p(\boldsymbol{w}|\mathcal{D}), \end{aligned}$$

where σ^2 is the homoscedastic measurement noise, and $p(\boldsymbol{w}|\mathcal{D})$ the posterior distribution of the parameters in the Bayesian neural network.

A forward pass through a BNN for a single input instance will result in a single output, just as for the deterministic FNN. Each input instance is fed through the network multiple times to obtain a posterior predictive distribution from which inference can be made. Each time the instance is forwarded through the network, a new set of parameters is sampled from their corresponding posterior distributions. Having a predictive distribution for all the instances being considers allows to reason about the model's predictive uncertainty and is useful for assessing how trustworthy the model is in making predictions.

The inference is based on the moments of the resulting predictive distribution, where the prediction itself is estimated using the expected value. Moreover, the uncertainty in the predictions is captured by the variance. The moments are estimated using Monte Carlo sampling.

$$\begin{aligned} \mathbb{E}[\hat{y}] &\approx \frac{1}{B} \sum_{i=1}^{B} f(\boldsymbol{x}, \boldsymbol{w}|\mathcal{D}), \\ \mathrm{Var}[\hat{y}] &\approx \sigma^2 + \frac{1}{B} \sum_{i=1}^{B} f(\boldsymbol{x}, \boldsymbol{w}|\mathcal{D})^2 - \mathbb{E}[\hat{y}]^2, \end{aligned}$$

where B is the number of Monte Carlo samples, $\mathbb{E}[\hat{y}]$ the expected value and $\mathrm{Var}[\hat{y}]$ the variance of the predictive distribution for $\boldsymbol{x}$. The first term in the predictive variance is the homoscedatic noise, representing the aleatoric uncertainty, whereas the second term represents the epistemic uncertainty. Note that for every Monte Carlo sample, a new set of parameters is sampled from their posterior distributions.

Some might argue that the major advantage of Bayesian methods is the ability to incorporate prior beliefs about the problem into the modelling process. In the case of Bayesian neural networks, the prior belief is incorporated into the prior distribution of the parameters of the network. As such, the ideal prior represents the expected model. However, due to a lack of interpretation of neural network models, it is impossible to get insight into how the model is supposed to be without first training the model. Setting the prior distribution for the parameters is a difficult task, as the parameters of a neural network offer no physical interpretation[6]. Incorporating

[6]except in the trivial case of a linear regression model, where the parameter for a variable represent the change in the output when changing the variable and keeping the remaining variables constant

prior beliefs about a system modelled by a neural network is thus a non-trivial task, and specifying meaningful prior distributions for the parameters in Bayesian neural networks is an active field of research [20, 36, 54, 59]. Moreover, the difficulty in specifying meaningful priors accounts for much of the criticism of Bayesian models [43].

Doing exact inference on BNNs is intractable. Moreover, traditional approximate inference methods such as MCMC do not scale well to the large number of parameters typically seen in neural networks applied in practical scenarios. Thus, most approaches to training BNNs are based on variational inference.

Chapter 3

Methodology

In the first two sections of this chapter, the different methods for constructing Bayesian neural networks are presented. In the subsequent section, we will describe how the predictive uncertainty is obtained and how to decompose the uncertainty into the different components described above. Further on, we will use the connection between Bayesian credible intervals and frequentist confidence intervals to numerically evaluate the resulting predictive uncertainty. In the last remaining sections of this chapter, we will describe how we will try to answer the research hypotheses regarding the epistemic uncertainty with respect to the amount of data and model complexity.

3.1 Monte Carlo Dropout

Monte Carlo Dropout, abbreviated as MC Dropout, is a rather simple method for estimating predictive uncertainty in deep neural networks. The method, as presented in the original article [17], requires no modification to the training routine nor the loss function and applies to any neural network architecture [61]. It can be seen as approximate Bayesian inference and does not, unlike other such methods [7, 45], impose additional computational cost during training [17].

The outline of the method is to turn on dropout during evaluation, i.e. when making predictions with the neural network, and perform B stochastic forward passes of the network using each input instance. By turning on dropout, each forward pass is analogous to sampling a neural network from a population containing all sub-networks that be formed by removing non-output neurons from the entire network [18], and forwarding the input instance through the sampled network. Sampling B neural networks and running multiple forward passes on the instance will result in a distribution of the prediction for the considered instance. Furthermore, the predicted value for the instance is estimated using the mean or mode of the predictive distribution, and the uncertainty is estimated using the sample variance. The method was first introduced in [16, 17], and further elaborated by one of the authors in [15] as part of his PhD thesis.

In [16, 17], the authors show that a deep neural network with arbitrary depth and non-linearities, with dropout applied before every non-output layer, is mathematically equivalent to approximate variational inference in the probabilistic deep Gaussian process. The deep Gaussian process was first introduced in [10] and is a deep belief network based on Gaussian processes, a Bayesian method providing uncertainty estimates to its predictions.

The connection between dropout in neural networks and the probabilistic deep

Gaussian process allows for new, probabilistic reasoning about the previously seen deterministic neural networks.

To obtain equivalence with the deep Gaussian process the models need to be specified correctly. The authors show that specifying a standard normal prior on the parameters in the network corresponds to adding a specific weight-decay parameter λ_i to the loss function being optimized during training.

$$\lambda_i = \frac{1 - p_i}{2N\tau}, \; i = 1, \ldots, L-1, \tag{3.1}$$

where p_i is the dropout probability of layer i, N the number of samples and $\tau > 0$ a model precision parameter. Specifying a more informative prior $p(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, l^{-2}I)$, given a prior length-scale l, requires scaling (3.1) by l^2 [16, p. 9].

It should now be clear that specifying a prior distribution on the parameters of the neural network introduce three additional hyper-parameters.

Table 3.1: Hyper-parameters introduced by the prior in MC Dropout.

Hyper-parameter	**Description**
Dropout probability p_i	Rate at which neurons in layer i are dropped
Model precision τ	Specifying the variability of the target variable
Prior length-scale l	Prior belief of the variance of the parameters

The precision parameter τ and prior length-scale l are determined by the following model assumptions.

$$\begin{aligned} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) &= \mathcal{N}(\hat{y}_i, \tau^{-2}I) \; i = 1, \ldots, n \\ p(\boldsymbol{w}) &= \mathcal{N}(0, l^{-2}I), \end{aligned}$$

where $\hat{y}_i = f(\boldsymbol{x}_i, \boldsymbol{w})$ is the predicted target variable given input instance $\boldsymbol{x}_i$ and parameters $\boldsymbol{w}$ for $i = 1, \ldots, n$. Note that $\boldsymbol{w} = \{W_1, \ldots, W_L\}$ is a vector containing the weight matrices for all the weight layers in the neural network.

The above model assumptions result in a homoscedastic model, where the hyper-parameter τ gives the constant aleatoric uncertainty. In this way, the practitioner is choosing the measurement noise. Because the level of measurement noise is typically unknown, it is desirable to have a model that can estimate it from data.

The original formulation of the MC Dropout model, as introduced by Gal and Ghahramani in [15, 16, 17], is extended to allow the aleatoric uncertainty to be estimated rather than chosen empirically by the practitioner. The idea builds upon the equivalence between training regression models using the Mean Squared Error (MSE) loss function and maximum likelihood (see section 2.3). We introduce a new loss function for training the MC Dropout models to estimate the aleatoric uncertainty, namely the negative log-likelihood (NLL).

First, the following model assumption is imposed.

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \mathcal{N}(\hat{y}_i, \sigma_i^2), \; i = 1, \ldots, N, \tag{3.2}$$

where the aleatoric uncertainty is represented by σ_i.

By assuming independece between the samples in the dataset $\mathcal{D}$, we can write the likelihood function of the above model (3.2) as

$$p(\mathcal{D}|X, \boldsymbol{w}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \prod_{i=1}^{N} \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i - \hat{y}_i}{\sigma_i}\right)^2}.$$

Furthermore, the log-likelihood is expressed as

$$\begin{aligned}\log p(\mathcal{D}|X, \boldsymbol{w}) &= \sum_{i=1}^{N} \log \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i - \hat{y}_i}{\sigma_i}\right)^2} \\ &= -\frac{N}{2} \log 2\pi - \sum_{i=1}^{N} \log \sigma_i - \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \cdot \frac{1}{\sigma_i^2}.\end{aligned} \tag{3.3}$$

The neural network is then trained by minimizing the negative log-likelihood function (3.3). This can be seen as a modified MSE loss function, where σ_i represents the aleatoric uncertainty.

For a heteroscedastic model with varying levels of uncertainty across samples, the NLL loss function is expressed as

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}, \boldsymbol{\sigma}) = \frac{N}{2} \log 2\pi + \sum_{i=1}^{N} \log \sigma_i + \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \cdot \frac{1}{\sigma_i^2}. \tag{3.4}$$

To obtain estimates of the aleatoric uncertainty in the heteroscedastic setting the neural network is made multi-headed, where for each instance $\boldsymbol{x}_i$, a forward pass through the network result in a prediction $\hat{y}_i$ and aleatoric uncertainty σ_i. The network is trained to output $\log \sigma_i^2$ rather than σ_i to avoid potential divisions by zero in (3.4). A computational graph of the multi-headed network is shown in Figure 3.1.

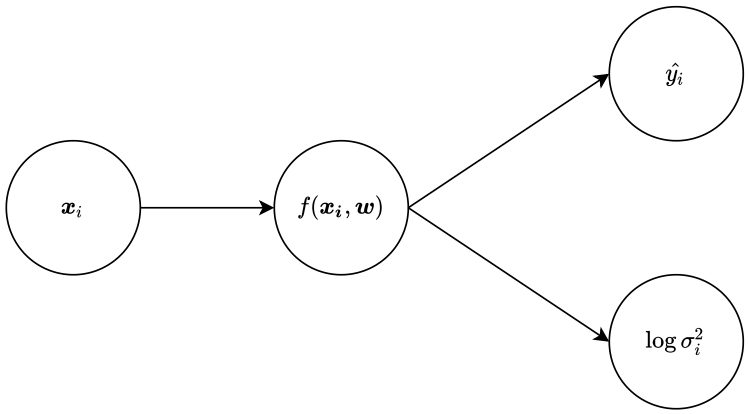


Figure 3.1: Computational graph of a multi-headed network $f(\cdot, \boldsymbol{w})$ with parameters $\boldsymbol{w}$ resulting in a prediction $\hat{y}_i$ and the aleatoric log-variance $\log \sigma_i^2$ for an instance $\boldsymbol{x}_i$.

For a homoscedastic model with constant noise levels, i.e. where $\sigma_i = \sigma \; \forall i$, the above NLL loss simplifies to

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}, \sigma) = \frac{N}{2} \log 2\pi\sigma^2 + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2. \tag{3.5}$$

In the simplified, homoscedastic case, the constant aleatoric uncertainty is treated as a parameter of the neural network model, rather than being predicted by the multi-headed network seen in Figure 3.1. The parameter is optimized alongside the weights and biases by minimizing the loss function (3.5).

In both cases, the aleatoric uncertainty is implicitly modelled using the negative log-likelihood loss function. We see that the squared error term $(y_i - \hat{y}_i)^2$ in the

loss functions forces the predictions to be close to the true targets, while the factor $\frac{1}{\sigma_i^2}$ enforces high uncertainty in the case of highly erroneous predictions. The $\log \sigma_i$ term acts as a constraint to the uncertainty, enforcing it not to grow indefinitely.

An overview of the method is described in Algorithm 2 below.

Algorithm 2: MC Dropout for a single input instance.

Inputs : Instance $\boldsymbol{x}$,
Trained neural network $f(\cdot, \boldsymbol{w})$ with depth L,
Number of stochastic forward passes B

Output: Predictive distribution $\hat{\boldsymbol{y}}(\boldsymbol{x})$ and aleatoric uncertainty $\sigma(\boldsymbol{x})$ for instance $\boldsymbol{x}$

for $b \leftarrow 1$ **to** B **do**
  sample $W_{i,b} \sim q(W_i)$, for $i = 1, \ldots, L$
  $\boldsymbol{w}_b = \{W_{1,b}, \ldots, W_{L,b}\}$
  $\hat{y}_b(\boldsymbol{x}),\ \log \sigma_b^2(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{w}_b)$
  $\sigma_b(\boldsymbol{x}) = \sqrt{e^{\log \sigma_b(\boldsymbol{x})}}$
end
$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \{\hat{y}_1(\boldsymbol{x}), \ldots, \hat{y}_B(\boldsymbol{x})\}$
$\sigma(\boldsymbol{x}) = \frac{1}{B} \sum_{i=1}^{B} \sigma_i(\boldsymbol{x})$

The approximating distribution of the weight matrices for each layer, $q(W_i)$, is defined in terms of the following [17].

$$\begin{aligned} W_i &= M_i \cdot \text{Diag}([Z_{i,j}]_{j=1}^{K_{i-1}}) \in \mathbb{R}^{K_{i-1} \times K_i)} \\ Z_{i,j} &\sim \text{Bernoulli}(1 - p_i), \quad \text{for } i = 1, \ldots, L-1,\ j = 1, \ldots, K_{i-1}, \end{aligned} \tag{3.6}$$

where $1 - p_i$ is the probability of retaining a hidden neuron in layer i with K_{i-1} input neurons and K_i output neurons. The matrix M_i contains the weights going in to layer i in the trained neural network. A similar calculation holds for the biases in the model.

Going through the calculations in (3.6), it is straightforward to see that the matrix W_i contains the weights going into layer i, with elements corresponding to the dropped neurons being zero.

An important hyper-parameter in MC Dropout is the dropout probability, i.e. the rate at which neurons in the non-output layers of the neural network is being dropped. In particular, because the approximate distribution of the weight matrices depends solely on the value of this parameter. Appropriately choosing the value of the dropout rate is thus essential to obtain reasonable estimates of the model uncertainty. The optimal values for hyper-parameters are typically found by doing a grid-search over the parameter space, fitting multiple models with a different set of parameters at each iteration, and choosing the parameters that optimize a chosen performance measure. However, one must act carefully when choosing the performance measure to optimize when tuning the value of the dropout rate in MC Dropout due to its probabilistic interpretation. A specific dropout rate found by a grid search for a given performance measure might not yield proper estimates of the model uncertainty. Hence, the value of the dropout rate is found empirically by trial and error in this thesis.

The variance of the predictive distribution using MC Dropout highly depends on the dropout rate, and it is important to choose it appropriately. This is also reflected in the approximate distribution of the weight layers (3.6). The higher the dropout

rate, the more likely it is to drop a neuron in a non-output layer of the neural network. This results in greater variability in the sampled networks, which in turn leads to greater prediction uncertainty. Furthermore, the neural networks become more likely to underfit the data using a high dropout rate, referring to dropout as a regularization technique.

A necessary condition for training a neural network with the MC Dropout method is that the model contains at least a single hidden layer. It is impossible to apply dropout to the input or the output variables without drastically changing the model. Applying dropout to the input, i.e. the explanatory variables in the model, may cause important variables to be dropped out in some iterations, while less important variables are dropped in others. This can be seen as an analogy to the variable selection provided by lasso-regularization [56], although in this case, the variables are dropped at random, and not determined by their importance to the regression.

If dropout is applied to the output, random parts of the output is dropped, and it is no longer possible to calculate the loss between the output and the ground truth value. Consequently, there is no value to propagate backwards through the neurons in the network during training, and the parameters are not updated.

3.2 Stochastic Gradient Variational Bayes

Stochastic Gradient Variational Bayes, abbreviated as SGVB and typically referred to as Bayes by backpropagation, is an algorithm for doing probabilistic backpropagation and thus training Bayesian Neural Networks. The algorithm was first proposed by a team from Google DeepMind in [7].

Unlike the deterministic approach to training neural networks, where backpropagation aims to obtain point estimates of the parameters, the probabilistic SGVB approach learns the variational parameters of a posterior probability distribution of the parameters. Consequently, the method requires the specification of a prior distribution $p(\boldsymbol{w})$ for each weight in the network. The algorithm updates the priors during training to obtain the corresponding posterior distributions.

Having a distribution for each parameter will, however, typically increase the number of learnable parameters, as in the case of a normally distributed posterior with parameters $\boldsymbol{\theta} = (\mu, \sigma)$ for every parameter of the model. Having distributions rather than point estimates for the parameters will, on the other hand, introduce the notion of uncertainty in the estimated parameters, which ultimately allows for assessing how trustworthy the model is in making predictions.

The algorithm is based on variational inference to obtain an approximate posterior distribution of the parameters in the neural network. Thus, the goal is to approximate the true posterior of the parameters $p(\boldsymbol{w}|\mathcal{D})$ using a variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$, and this is achieved by minimizing the KL-divergence between the two distributions.

As derived in section 2.4.1, the KL-divergence between the variational distribution and the true posterior can be expressed as follows.

$$\begin{aligned} D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D})] &= D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w})] - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log p(\mathcal{D}|\boldsymbol{w})] \\ &\quad + \log p(\mathcal{D}) \\ &= \log p(\mathcal{D}) - \mathcal{L}(q_{\boldsymbol{\theta}}(\boldsymbol{w})) \end{aligned}$$

We want to minimize the KL-divergence between $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ and $p(\boldsymbol{w}|\mathcal{D})$, and the above expression motivates the following cost function.

$$J(\boldsymbol{w}, \boldsymbol{\theta}) = D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w})] - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log p(\mathcal{D}|\boldsymbol{w})] = -\mathcal{L}(q_{\boldsymbol{\theta}}(\boldsymbol{w})), \tag{3.7}$$

where $\boldsymbol{w}$ is a vector of parameters for a certain layer in the neural network, and $\boldsymbol{\theta}$ the variational parameters specifying the properties of the variational distirbution. We can see that by minimizing the cost function $J(\boldsymbol{w}, \boldsymbol{\theta})$ we are simultaneously maximizing the ELBO function $\mathcal{L}(q_{\boldsymbol{\theta}}(\boldsymbol{w}))$ (see section 2.4.1), which in turn minimize the KL-divergence between $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ and $p(\boldsymbol{w}|\mathcal{D})$.

The terms in the above cost (3.7) represents a trade-off between the complexity of the data given by the log-likelihood $\log p(\mathcal{D}|\boldsymbol{w})$, and the typically simple prior $p(\boldsymbol{w})$ [7]. By minimizing the cost, we want the resulting distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ to capture the complexity of the data given by the likelihood while simultaneously being similar to the prior. The similarity with the prior is reflected in the KL-divergence, capturing the similarity between the two distributions.

Similar to the cost functions for the MC Dropout models (see section 3.1), the above cost (3.7) contains a negative log-likelihood term. The difference lies in the additional KL-divergence term acting as a regularizer in the SGVB cost. Moreover, we calculate the expected log-likelihood over the variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$.

Using the definition of the KL-divergence (2.15) we can expand the cost in terms of expectations.

$$J(\boldsymbol{w}, \boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{w})}{p(\boldsymbol{w})}\right] - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log p(\mathcal{D}|\boldsymbol{w})]$$

$$= \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log q_{\boldsymbol{\theta}}(\boldsymbol{w})] - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log\left(p(\boldsymbol{w})p(\mathcal{D}|\boldsymbol{w})\right)].$$

The expected values are estimated using Monte Carlo sampling, and the exact cost is approximated as

$$\begin{aligned} J(\boldsymbol{w}, \boldsymbol{\theta}) &\approx \frac{1}{N}\sum_{i=1}^{N} \log q(\boldsymbol{w}^{(i)}|\boldsymbol{\theta}) - \log p(\boldsymbol{w}^{(i)}) - \log p(\mathcal{D}|\boldsymbol{w}^{(i)}) \\ &= \frac{1}{N}\sum_{i=1}^{N} j(\boldsymbol{w}^{(i)}, \boldsymbol{\theta}), \end{aligned}$$

where $\boldsymbol{w}^{(i)}$ is the i'th Monte Carlo sample drawn from the variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$, for $i = 1, \ldots, N$, where N denotes the number of Monte Carlo samples.

We assume that the variational distribution is normally distributed with mean μ and standard deviation σ. Rather than sampling the parameters directly from the variational distribution, we can sample from parameter-free noise and scale it appropriately to obtain a sample of the parameters. Doing backpropagation through stochastic neurons in a neural network, in which the parameters represents in this case, is not possible. However, this reparametrization trick [33] allows to do backpropagation through the network, as we the stochastic part is moved outside the neuron. The reparametrization trick is carried out as follows.

$$\boldsymbol{w} = \mu + \sigma \odot \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, I) \tag{3.8}$$

A schematic view of the above reparametrization trick is seen in Figure 3.2.

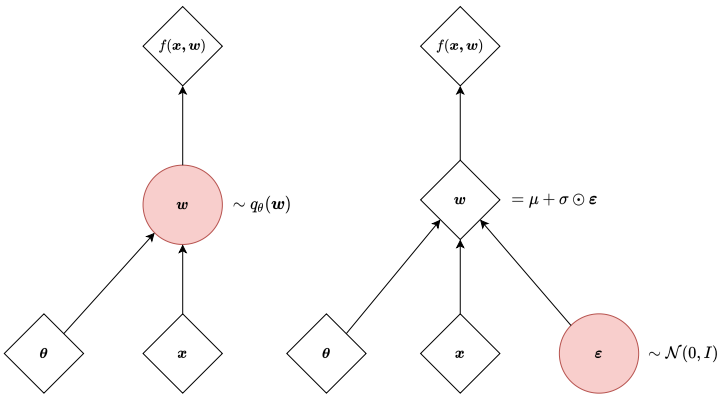


Figure 3.2: Original (left) and reparametrized form (right) of the sampling procedure for the paramters $\boldsymbol{w}$ in the Bayesian neural network $f(\cdot, \boldsymbol{w})$ for an instance x. The white squares represent deterministic neurons, while the stochastic neurons are shown as red circles. Adapted from [34].

In [32] the authors propose a new local reparametrization trick that is shown to reduce the variance of the stochastic gradients in BNNs. In this thesis, we will however stick to the above reparametrization trick (3.8) due to its simplicity.

The variational parameters $\boldsymbol{\theta} = (\mu, \sigma)$ are found using gradient descent during training. The variance is required to be non-negative by definition, and the standard deviation is parametrized as $\sigma(\rho) = (1 + e^{\rho})$.

Algorithm 3 describes the SGVB algorithm for a single iteration of the optimization, where the variational parameters are $\boldsymbol{\theta} = (\mu, \rho)$.

Algorithm 3: SGVB for a single iteration of the optimization. Adapted from [7].

Result: $\boldsymbol{\theta} = (\mu, \rho)$
Sample $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$
Let $\boldsymbol{\theta} = (\mu, \rho)$
Let $\boldsymbol{w} = \mu + \log(1 + e^{\rho}) \odot \boldsymbol{\varepsilon}$
Let $j(\boldsymbol{w}, \boldsymbol{\theta}) = \log q_{\boldsymbol{\theta}}(\boldsymbol{w}) - \log p(\boldsymbol{w}) - \log p(\mathcal{D}|\boldsymbol{w})$
Calcualte gradients with respect to the variational parameters

$$\nabla_{\mu} j(\boldsymbol{w}, \boldsymbol{\theta}) = \frac{\partial j(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{w}} + \frac{\partial j(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \mu}$$

$$\nabla_{\rho} j(\boldsymbol{w}, \boldsymbol{\theta}) = \frac{\partial j(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{w}} \frac{\boldsymbol{\varepsilon}}{1 + \exp(\rho)} + \frac{\partial j(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \rho}$$

Update the parameters (gradient descent step)

$$\mu \leftarrow \mu - \alpha \nabla_{\mu} j(\boldsymbol{w}, \boldsymbol{\theta})$$

$$\rho \leftarrow \rho - \alpha \nabla_{\rho} j(\boldsymbol{w}, \boldsymbol{\theta})$$

Note that the gradient of the cost w.r.t the variational parameters depends on the same term, namely the partial derivative of the cost w.r.t the parameters, $\frac{\partial j(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{w}}$. These are the same gradients we obtain by doing standard backpropagation on a deterministic neural network. The remarkable insight from this result is that one can update the variational parameters, thus learning the mean and standard deviation of the variational distribution simply by calculating gradients using standard backpropagation and scaling them as in the above algorithm.

Similar to the MC Dropout models, we will model the aleatoric uncertainty in a homoscedastic and heteroscedastic manner by setting up a mathematical model that will be specified in Section 4.4. This is done by using the following expressions for the log-likelihood in (3.7).

$$\log p(\mathcal{D}|\boldsymbol{w})_{\text{Homoscedastic}} = \frac{N}{2} \log 2\pi\sigma^2 + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i, \boldsymbol{w}))^2,$$

$$\log p(\mathcal{D}|\boldsymbol{w})_{\text{Heteroscedastic}} = \frac{N}{2} \log 2\pi + \sum_{i=1}^{N} \log \sigma_i + \frac{1}{2} \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i, \boldsymbol{w}))^2,$$

where y_i is the ground truth target value for the prediction $f(\boldsymbol{x}_i, \boldsymbol{w})$ for instance $\boldsymbol{x}_i$, and σ_i the aleatoric uncertainty. For the homoscedastic model we have $\sigma_i = \sigma \; \forall i$.

3.3 Quantifying Predictive Uncertainty

The predictive uncertainty is estimated using a sampling approach. More specifically by performing B stochastic forward passes through the network using the same input instance and then repeating the process for all instances in the test set. This will result in a posterior predictive distribution for all input instances, and the uncertainty of the predictions are measured using the standard deviation of the distribution. The obtained uncertainty estimates are a combination of the aleatoric and epistemic uncertainty components, and we will see how to decompose the predictive uncertainty into these components in section 3.3.1.

The predicted target variable is obtained using a linear activation function in the final layer of the network. Every time an input instance is passed through the network, the network will output a single value for the predicted target value for that particular instance. By passing a single input instance through the network multiple times, we obtain a predictive distribution for that instance. This is, of course, on the premise that we are working with a probabilistic Bayesian neural network. A deterministic neural network will output the same value for a given input instance if forwarded through the network multiple times.

Having a predictive distribution for each input instance allows one to reason about the uncertainty of the network predictions. By estimating the predicted value as the mean of the predictive distribution, we can construct credible intervals around the predicted value by calculating the standard deviation of the predictive distribution. We can then use these credible intervals to assess the uncertainty of the predictions, using the fact that a broader credible interval corresponds to greater uncertainty in the estimated value the interval is constructed around.

Assuming the predicted target variable is normally distributed with mean μ and standard deviation σ, we can construct the credible interval as

$$[\mu - z_{\frac{\alpha}{2}}\sigma,\ \mu + z_{\frac{\alpha}{2}}\sigma], \tag{3.9}$$

where $z_{\frac{\alpha}{2}}$ is the $\frac{\alpha}{2}$ percentile of the standard normal distribution. In this case, setting $z_{\frac{\alpha}{2}} = 1.96$ yields a 95% credible interval.

The performance of the models are measured using the Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics on the test set, where a lower value corresponds to better predictive performance. The metrics are defined as

$$\begin{aligned} MSE &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \\ MAE &= \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \end{aligned} \tag{3.10}$$

where N is the size of the test dataset, y_i the ground truth and $\hat{y}_i$ the predicted target value for instance $i = 1, \ldots, N$.

Following the same sampling approach as described above, we can obtain distributions for the metrics (3.10). This allows to construct credible intervals for the performance metrics, and the results are presented as follows.

$$\begin{aligned} &\mu_{MSE} \pm \sigma_{MSE}, \\ &\mu_{MAE} \pm \sigma_{MAE}, \end{aligned} \tag{3.11}$$

where $\mu_{MSE,MAE}$ and $\sigma_{MSE,MAE}$ are the mean and standard deviation of the distribution of the performance metrics.

3.3.1 Decomposing the Uncertainty

We have already seen that the predictive uncertainty can be decomposed into two inherently different components, namely the aleatoric and epistemic uncertainty.

The aleatoric uncertainty is concerned with the uncertainty related to the data-generating process, and can e.g. represent measurement noise and inherent variability in the data. In contrast, the epistemic uncertainty is related to a lack of knowledge about which model generated the observed data and accounts for the uncertainty in the parameters of the model [30].

We estimate the aleatoric uncertainty in two different settings, obtaining a homoscedastic and heteroscedastic aleatoric uncertainty. Implementing a homoscedastic model is quite simple, but doing so may cause a loss of important information about the total predictive uncertainty as the aleatoric uncertainty is potentially over- and under-estimated in different intervals of the test set. On the contrary, a heteroscedastic model provides a richer representation of the predictive uncertainty. However, it is harder to implement.

In both situations, the aleatoric uncertainty is implicitly estimated using the negative log-likelihood loss function. In the heteroscedastic setting, the aleatoric uncertainty is estimated alongside the target variable using a multi-headed network. For the homoscedastic models, the aleatoric uncertainty is treated as a model parameter optimized alongside the parameters of the neural network by minimizing the loss function. Each setting corresponds to a slightly different loss function, and the two loss functions are expressed as follows.

$$\mathcal{L}_{\text{Heteroscedastic}}(\boldsymbol{y}, \hat{\boldsymbol{y}}, \boldsymbol{\sigma}) = \frac{N}{2}\log 2\pi + \sum_{i=1}^{N}\log\sigma_i + \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \cdot \frac{1}{\sigma_i^2}$$

$$\mathcal{L}_{\text{Homoscedastic}}(\boldsymbol{y}, \hat{\boldsymbol{y}}, \sigma) = \frac{N}{2}\log 2\pi\sigma^2 + \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,$$

where we see that the homoscedastic loss is a special case of the heteroscedastic loss with $\sigma_i = \sigma\ \forall i$. Note however that the loss functions are slightly different for the SGVB method, where the difference lies in an additional KL-divergence term (see section 3.2).

The epistemic uncertainty relates to the uncertainty in the parameters of the model and is captured by the standard deviation of the posterior predictive distribution for each input instance. In each forward pass of the network, a set of model parameters are sampled from their corresponding (approximate) posterior distributions. Forwarding the same input instance multiple times is equivalent to using multiple models to predict the target value. We can thus use the variance, or standard deviation, of the predictive distribution for each instance as a measure of the epistemic uncertainty [30].

Having obtained the aleatoric and epistemic uncertainty, the total predictive variance is calculated as

$$\sigma^2_{\text{Predictive}} = \sigma^2_{\text{Aleatoric}} + \sigma^2_{\text{Epistemic}}, \tag{3.12}$$

where $\sigma^2_{\text{Aleatoric}}$ and $\sigma^2_{\text{Epistemic}}$ is the aleatoric and epistemic variance, respectively.

The algorithm for obtaining the predictive variance as well as the aleatoric and epistemic components is summarized in Algorithm 4, showing the formulas for obtaining the components of the total predictive uncertainty (3.12).

Algorithm 4: Calculating predictive uncertainty for a single input instance.

Inputs : Instance $\boldsymbol{x}$,
Trained neural network $f(\cdot, \boldsymbol{w})$ with depth L,
Number of stochastic forward passes B

Output: Predictive variance $\sigma^2_{\text{Predictive}}(\boldsymbol{x})$

for $b \leftarrow 1$ **to** B **do**
  Sample $W_{i,b} \sim q(W_i),\ i = 1, \ldots, L$
  $\boldsymbol{w}_b = \{W_{1,b}, \ldots, W_{L,b}\}$
  $\hat{y}_b(\boldsymbol{x}),\ \log \sigma^2_{\text{Aleatoric},b}(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{w}_b)$
  $\sigma^2_{\text{Aleatoric},b}(\boldsymbol{x}) = e^{\log \sigma^2_{\text{Aleatoric},b}(\boldsymbol{x})}$
end
$\sigma^2_{\text{Aleatoric}} = \frac{1}{B} \sum_{i=1}^{B} \sigma^2_{\text{Aleatoric},i}(\boldsymbol{x})$
$\sigma^2_{\text{Epistemic}}(\boldsymbol{x}) = \text{Var}[\hat{\boldsymbol{y}}(\boldsymbol{x})] = \frac{1}{B} \sum_{i=1}^{B} \hat{y}_i(\boldsymbol{x})^2 - \frac{1}{B} \sum_{i=1}^{B} \hat{y}_i(\boldsymbol{x}) \frac{1}{B} \sum_{i=1}^{B} \hat{y}_i(\boldsymbol{x})$
$\sigma^2_{\text{Predictive}}(\boldsymbol{x}) = \sigma^2_{\text{Aleatoric}}(\boldsymbol{x}) + \sigma^2_{\text{Epistemic}}(\boldsymbol{x})$

The uncertainty is presented as credible intervals around the predictions for all the instances in the test set. To separate the aleatoric and epistemic uncertainty, we construct credible intervals based on the total predictive uncertainty and the epistemic uncertainty alone. The differences between these credible intervals will then account for the aleatoric uncertainty in the data.

3.4 Quantitatively Evaluating the Predictive Uncertainty

If we want to use the estimated uncertainty of a probabilistic model to aid decision-making, we want the provided uncertainty to be estimated appropriately. Too high uncertainty estimates are evidence of an under-confident model, whereas too low uncertainty estimates indicate that the model is over-confident. Either case is unwanted, and we want a model that provides predictions with the right amount of credibility. In that way, we can trust the uncertainty estimates and safely use the model in a decision-making process.

An over-confident model may yield confident yet erroneous predictions that are potentially harmful depending on the context given by the decision-making process. On the other hand, an under-confident model may yield highly uncertain yet correct predictions. A potential use case for the probabilistic models is to trust the predictions if they have a predictive uncertainty below some threshold and pass the prediction to human inspection if the uncertainty is above the specified threshold. So in the case of an under-confident model, the predictions will potentially be passed to a human for further inspection more rapidly than if the uncertainty is appropriately estimated, even if the prediction turns out to be correct. This will, in turn, slow down the automation process. On the contrary, the potentially erroneous predictions of an over-confident model may have a predictive uncertainty approved by the uncertainty threshold. In this hypothetical case, the prediction is then passed on without human inspection, and the consequences are potentially harmful.

To evaluate the uncertainty estimates, we use the connection between the Bayesian credible interval and the frequentist confidence interval, as presented by Berger and described in section 2.4. This is obtained by constructing calibration curves [37], where the empirical coverage probability of the credible interval is plotted against the significance level of the corresponding confidence interval. In a regression setting, calibration refers to the expected fraction of samples that fall inside a credible interval around the prediction for a given probability level. More specifically, it means that we should expect 95% of the ground-truth values of the target in the test set to be inside a 95% credible interval around the corresponding predictions. This expected behaviour is achieved for a perfectly calibrated model for all probability levels in the range $0 - 100\%$.

The evaluation criteria depend on the empirical coverage probability, which is the fraction of samples in the test set inside the credible interval for a given probability level. We define the empirical coverage as

$$C_e = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1} \left[L(\hat{y}_i) < y_i < U(\hat{y}_i) \right], \tag{3.13}$$

where $\mathbb{1}[\cdot]$ is the indicator function, $L(\hat{y}_i)$ and $U(\hat{y}_i)$ the lower and upper bound of the credible interval for prediction $\hat{y}_i$ at a certain probability level and y_i the true target value for sample $i = 1, \ldots, N$ in the test set.

Note that when describing a credible interval around a prediction, we are talking about the probability that the true value is inside the interval. This probability is referred to as the *probability level* of the credible interval. On the other hand, for the frequentist confidence interval, we are talking about a *significance level*, representing the long-term frequency of the confidence interval containing the true value.

By constructing a set of credible intervals (3.9) using different probability levels and calculating the empirical coverage (3.13), we can construct calibration curves by plotting the empirical coverage against the significance level for the corresponding confidence intervals. This will result in a diagonal line with a unit slope for a perfectly calibrated model. Furthermore, we can assess whether the model is under or over-confident by looking at the potential discrepancy from the diagonal line. If the empirical coverage is above the diagonal line, there are a greater fraction of samples inside the credible interval than expected by the significance level. The credible interval is thus broader than expected, and the uncertainty is over-estimated. This implies that the model is under-confident, as the uncertainty is greater than expected. On the contrary, an over-confident model has tighter credible intervals. This will result in fewer samples falling inside the interval than what is expected from the significance level, such that the empirical coverage is less than optimal. As a result, the calibration curve lies below the diagonal.

We will see in section 4.2 that the data is structured in a way that allows the construction of multiple calibration curves for each model. To summarize the calibration curves over the test set, we will present the results as the mean empirical coverage for a given significance level, and provide a 95% credible interval around this mean for all significance levels in the range $10 - 100\%$. In this way, we can reason about the uncertainty of the uncertainty estimates of the models and use that in conjunction with the mean calibration to evaluate the uncertainty estimates.

3.4.1 Effect of Modelling Aleatoric Uncertainty

To look into the effect of modelling the aleatoric uncertainty, we will construct calibration curves based on the epistemic uncertainty estimates alone. By looking at how the calibration curves differ from the calibration curves based on the total predictive uncertainty, we can study the effect of modelling and including the aleatoric uncertainty component to the total predictive uncertainty of the models. The differences can also be used as a proxy for how well the aleatoric uncertainty is estimated.

3.5 Epistemic Uncertainty and Training Set Size

The epistemic uncertainty refers to the lack of knowledge of the process being modelled, and can principally be reduced by collecting additional information, e.g. data [27]. This lack of knowledge is represented in the uncertainty of the parameters of the model.

By training several models on a set of fractions of the entire training set, we can empirically study how the epistemic uncertainty evolves with the amount of data the models are allowed to process during training. More specifically, the approximation uncertainty. As stated in section 1.5, we have not seen any experimental nor theoretical justification of this stated property of the uncertainty component.

We consider a set of fractions from $10 - 100\%$ of the entire training set, meaning that for every model we consider, we will have to re-train the model multiple times using a different training set. The training sets are constructed by randomly sampling a fraction of the entire training set for all the fractions we consider. Further on, we train our models for all the fractions and make predictions on the entire test set. It is important to note that the size of the test set is constant across all fractions of the training set.

The uncertainty is quantified and decomposed as described in section 3.3.1. For every sample in the test set, we obtain a predictive distribution with variance determined by the sum of the epistemic and aleatoric variances. To extract the epistemic uncertainty, we isolate the epistemic variance and calculate the standard deviation by taking the square root. All the samples in the test set are now equipped with epistemic uncertainty in terms of a standard deviation. To summarize the epistemic uncertainty over the entire test set, we report the epistemic uncertainty as the mean standard deviation over all samples in the test set. The procedure is summarized in Algorithm 5.

Referring back to Section 3.1, the posterior predictive distribution of the parameters in a BNN constructed with the MC Dropout method depends strongly on the dropout rate. Consequently, the epistemic uncertainty of the MC Dropout models is directly related to the dropout rate. To study the effect of the dropout rate on the epistemic uncertainty of these models, additional models using different dropout rates are fitted. To isolate the potential effect of the dropout rate, all the other hyper-parameters and architectural considerations remain fixed.

Algorithm 5: Epistemic uncertainty for a fraction of the training set.

Inputs : Training set $\mathcal{D}_{\text{Train}}$,
Test set $\mathcal{D}_{\text{Test}}$,
Number of stochastic forward passes B,
Fraction Φ of training set

Output: Epistemic uncertainty $\sigma_{\text{Epistemic}}$ for fraction f of the dataset

Sample fraction Φ of $\mathcal{D}_{\text{Train}}$
Train neural network $f(\cdot, \boldsymbol{w})$ using down-sampled training set
for $\boldsymbol{x}_j \in \mathcal{D}_{Test},\ j = 1, \ldots, N_{Test}$ **do**
  for $b \leftarrow 1$ **to** B **do**
    Sample $W_{k,b} \sim q(W_k),\ k = 1, \ldots, L$
    $\boldsymbol{w}_b = \{W_{1,b}, \ldots, W_{L,b}\}$
    $\hat{y}_b(\boldsymbol{x}_j) = f(\boldsymbol{x}_j, \boldsymbol{w}_b)$
  end
  $\sigma^2_{\text{Epistemic}}(\boldsymbol{x}_j) = \text{Var}[\hat{\boldsymbol{y}}(\boldsymbol{x}_j)]$
  $\sigma_{\text{Epistemic}}(\boldsymbol{x}_j) = \sqrt{\sigma^2_{\text{Epistemic}}(\boldsymbol{x}_j)}$
end
$\sigma_{\text{Epistemic}} = \frac{1}{N_{\text{Test}}} \sum_{j=1}^{N_{\text{Test}}} \sigma_{\text{Epistemic}}(\boldsymbol{x}_j)$

3.6 Epistemic Uncertainty and Model Complexity

We have already seen how we can study epistemic uncertainty by training models with training sets of different sizes. However, this approach can only tell us something about the approximation component of the epistemic uncertainty, i.e. the discrepancy between the fitted model and the optimal model within the given hypothesis space.

The model uncertainty represents how suitable the chosen hypothesis space is for explaining the data at hand and corresponds to the discrepancy between the ground-truth and fitted model. By setting up a hypothesis space that is far away from containing a proper model, the model uncertainty will be high. On the other hand, setting up a proper hypothesis space will resolve this uncertainty.

We will study the model uncertainty by fitting multiple models by setting up different hypothesis spaces and observing how well the different models can explain the epistemic uncertainty. We will train the different models using a constant dataset size to avoid interference with the approximation uncertainty. More complex models require more data, and we can still expect the approximation uncertainty to increase when increase model complexity. However, in large sample situations, we can expect the potential differences in the epistemic uncertainty to result from the model uncertainty when using a dataset with a constant size across multiple model complexities.

The different hypothesis spaces are represented by the complexities of the neural network models, and we will consider four different complexities. The different complexities correspond to a linear model, a single-layer model, an intermediate model with two hidden layers and a complex model with ten hidden layers. The linear model can be formulated as a neural network with no hidden layers, i.e. a direct connection between the input and output variables. Referring back to Section 3.1, a model trained with the MC Dropout method needs to contain at least a single hidden layer. Consequently, it is impossible to train a linear model with MC Dropout, and the method is omitted from this analysis. Thus, all the considered models are trained using the SGVB method, where the aleatoric uncertainty is modelled in a homoscedastic and heteroscedastic setting. This accounts for fitting two models for each model complexity, i.e. eight models in total. The architectural configuration of the models are presented in section 4.4.

To present the epistemic uncertainty, the predictive uncertainty is quantified and decomposed according to Algorithm 4 in section 3.3.1 for both SGVB models. The epistemic uncertainty is aggregated into a single number by computing the mean epistemic uncertainty for all the samples in the test set. The results are presented as a plot of the aggregated epistemic uncertainty against model complexity.

Chapter 4

Experimental Setting

This masters thesis is a collaboration with the Norwegian Exploration and Production (E&P) company Aker BP. The company provides a problem and a dataset for solving a regression task, and the main objective of this thesis is to estimate and analyze the uncertainty provided by neural network models solving the task.

The data is gathered in boreholes while drilling the subsurface and consists of measurements of different geophysical properties. The problem is concerned with predicting the acoustic log for S-waves, given a set of geophysical measurements representing subsurface properties.

This chapter aims at giving the reader a motivation behind the problem provided by the dataset before describing the dataset in detail. Furthermore, the results of a preliminary analysis of the dataset are presented. The remaining sections of this chapter revolve around describing the mathematics and architectural configuration of the models and the deep learning framework for developing the models.

4.1 Motivation

The acoustic logs are used for several applications, including lithology interpretation, seismic depth conversion and direct hydrocarbon indication. Two kinds of acoustic logs are measured, namely compressional and shear wave acoustic logs. Because fluids lack shear strength, the shear waves will not propagate through formations containing fluids. By looking at the difference between the measured compressional and shear acoustic logs, one can infer whether or not the rock formations adjacent to the measurements contain fluids. One can use this in conjunction with other logs and analyses to check whether these fluids are hydrocarbons or water.

The acoustic measurements are costly to perform and are typically only carried out in areas with high human confidence of containing hydrocarbons. By doing so, one might miss out on unexpected reservoirs containing oil and gas. Hence, it is beneficial to have a complete picture of the acoustic logs over the whole borehole, independent of whether a human finds it likely or not to contain hydrocarbons. By training a machine learning algorithm using the costly measurements, one can apply this model and make predictions in the areas where one otherwise would not perform measurements, e.g. due to low confidence in containing hydrocarbons. It is too expensive to measure the acoustics logs along the entire borehole, so having a model that can provide reliable measurement predictions is very useful.

Moreover, measurements may be unavailable due to drill technical reasons. It is thus favourable to have a model that can predict the missing measurements. Having a measure of uncertainty attached to the predictions makes the model even more valuable, as one can choose whether or not to discard the predictions based on the level of uncertainty. Besides, having a good model for the acoustic log can potentially supersede the need for an instrument, which is costly to use, maintain and repair.

4.2 Data

A description of the problem and dataset provided by Aker BP is given in this section. The preprocessing and feature engineering schemes applied to make the data suitable for modelling are described following the dataset's description and an exploratory data analysis.

The dataset is based on open wireline logs, where different geophysical properties are measured in a borehole using different instruments attached to the borehead. Thus, the measurements are intrusive, as one needs to bore down into the subsurface to obtain the measurements.

The data is gathered from 34 wells from offshore Norway, and the well logs include the depth of measurements, acoustic S-wave log and other logs measuring physical properties of the formation. A map showing the location for a subset of the wells are shown in Figure 4.1 below, where the wells are marked as red circles. Existing petroleum fields are marked as green shapes, and fault lines are marked in black.

Initially, the raw dataset contains 44 features. The majority of these features are geophysical logs, while some are flags for imperfect measurements and corrected versions of other logs. However, not all of these are relevant for the modelling. A list of the variables included in the modelling, as well as a brief description, is found in Table 4.1 below. The variables are chosen based on the same set of variables used in human interpretation of the data.

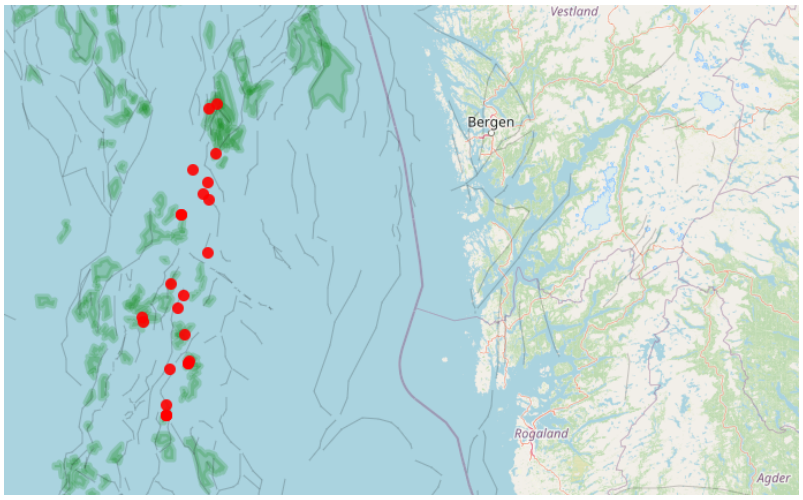


Figure 4.1: Map of some of the wells (red) in the dataset. The green shapes represents existing petroleum fields, and the black lines are fault lines. The purple line outside the coast of Norway is the norwegian territorial border.

Table 4.1: Variables in the dataset with corresponding units and descriptions.

Variable	**Unit**	**Description**
ACS	µs/ft	S-wave travel time log (target variable)
AC	µs/ft	P-wave travel time log
AI	MPa s/m	Acoustic impedance log
BS	inches	Bit size log, measuring the size of the bore head
CALI	inches	Caliper, measuring the width of the borehole
DEN	g/m^3	Density log
GR	gAPI	Gamma ray log
NEU	1 (m^3/m^3)	Neutron porosity log
RMED	Ω m	Medium depth electrical resistivity log
DEPTH	m	Depth of measurement

A brief description of how the different measurements are carried out follows. The material is based on the self-learning module on open hole wireline logging by Tracs International [58].

The sonic logs, i.e. ACS and AC, measure acoustic pulses' travel times against depth through formations close to the borehole. The measurements are obtained by measuring the arrival times between the source of the acoustic pulse and two receivers spaced at different distances from the transmitter. By subtracting the travel times to the nearest from the farthest receiver and dividing by the distance between the two, one obtains the acoustic velocity over the interval between the receivers. The ACS log measures the travel times for the shear waves (S-waves), while the AC log measures the travel times for the compressional waves (P-waves) of the acoustic signal. In dry rocks, these logs are typically strongly correlated. However, in the presence of fluids in the pores of the rock, e.g. water, oil and gas, the S-wave velocity vanishes due to the lack of shear strength. The difference between these logs is thus very useful for exploring the subsurface for oil and gas.

Acoustic impedance, AI, is simply the product of the bulk density and the seismic velocity [48] in the rocks adjacent to the borehead. The log contains important information about the nature of the rock and variations in lithology [3].

The logs BS and CALI represent the borehole's width in terms of the diameter, and they are both measured in inches. The bit size log is the actual width of the borehead, while the caliper log measures the width of the resulting borehole mechanically. These logs will typically correlate strongly. In loose rocks, the caliper will typically be greater than the bit size. This is because loose rocks are less capable of keeping intact after drilling, making the borehole less stable.

The bulk density of the rock formation in the subsurface is measured using a radioactive source that emits medium-energy gamma rays. When radiating through the formation, the gamma rays collide with electrons and are scattered. Ultimately, the gamma rays are detected at a fixed distance from the transmitter, and the number of detected gamma rays are counted. The count is inversely proportional to the electron density, which is related to the bulk density of the formation.

The gamma-ray log, GR, represents the amount of natural emission of gamma rays from the rock formation adjacent to the borehole. GR levels are measured in gAPI, a unit of radioactive emission of gamma rays. The natural emission is a result of decaying radioactive isotopes naturally occurring in the rock formation. In a petroleum field, the oil and gas are contained in a reservoir rock's pore volume, which is typically sandstone. Naturally, because of capillary forces and pressure gradients, the petroleum rises towards the surface, trying to escape the reservoir rock. The petroleum is then trapped in the reservoir in the presence of a cap rock overlying the reservoir rock, which is typically shale. Because shales and sandstones have notably different gamma-ray signatures [49], the gamma-ray log is particularly useful in the exploration phase of a petroleum reservoir.

The neutron porosity log NEU represents the porosity of the rock formation. It is measured by emitting high energy neutrons into the formation and detecting the density of neutrons at a fixed distance from the transmitter. The emitted neutrons are slowed down and captured primarily by collisions with hydrogen atoms in the rock formation, and the remaining neutrons are detected at the receiver. Hydrogen atoms are predominantly present in water and hydrocarbons in the pore volume of the rock, thus relating the neutron porosity to the porosity of the rock. The neutron porosity is dimensionless, as it represents the fraction of pore volume to the total volume of the rock.

The medium depth resistivity log, RMED, measures the electrical resistivity of the rock formation at a medium distance from the borehole horizontally into the rock. Resistivity logging in boreholes is usually concerned with three depths of measurements, and the medium reading resistivity log considers the intermediate invasion depth. Resistivity logs are useful because, unlike water, hydrocarbons in the pore volume of the rock does not conduct electricity [50]. As a result, resistivity logs will be high in the presence of hydrocarbons and low in the presence of water. One can thus use resistivity logs to separate hydrocarbons and water present in the rock formations.

The DEPTH log is a measurement of the length of the wellbore along its path. This measurement will thus differ from the True Vertical Depth (TVD), measuring the vertical displacement from the seabed surface to the depth of measurement. However, if the path of the borehole is truly vertical, the depth log and TVD will coincide perfectly.

The problem aims at predicting the numerical value of the acoustic log for the S-waves in the subsurface, given a set of well-log measurements taken at 34 different wells in the North sea. We are thus concerned with a regression-type problem. The target variable for the regression is ACS, and the remaining variables are used as

explanatory variables. The distribution of the target variable is shown in Figure 4.2. All the following distributions are shown as histograms with a kernel density estimate for the corresponding continuous distributions.

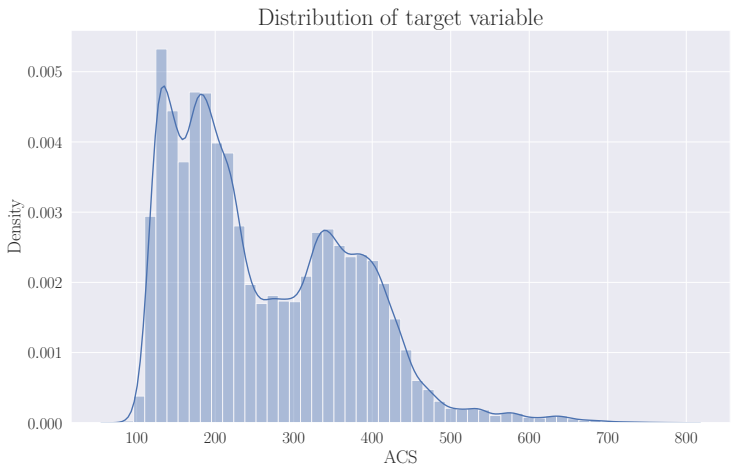


Figure 4.2: Distribution of the target variable in the dataset.

The data is gathered from 34 different wells from offshore Norway, and the different wells differ in terms of location and depths. As a result, the measurements are likely to differ across wells as they are taken in potentially widely different geological formations. A plot of a random subset showing four wells is shown in Figure 4.3. The name of each well is specified in the title of each subplot.

Figure 4.3: Distribution of the target variable for a random subset of 4 wells in the dataset. The name of the wells are specified in the title of each subplot.

It is clear from Figure 4.3 above that the distribution of the target variable differs across the wells in the dataset. This needs to be taken into account further down the modelling pipeline.

A plot of the distribution of the explanatory variables in the dataset is shown in Figure 4.4 below.

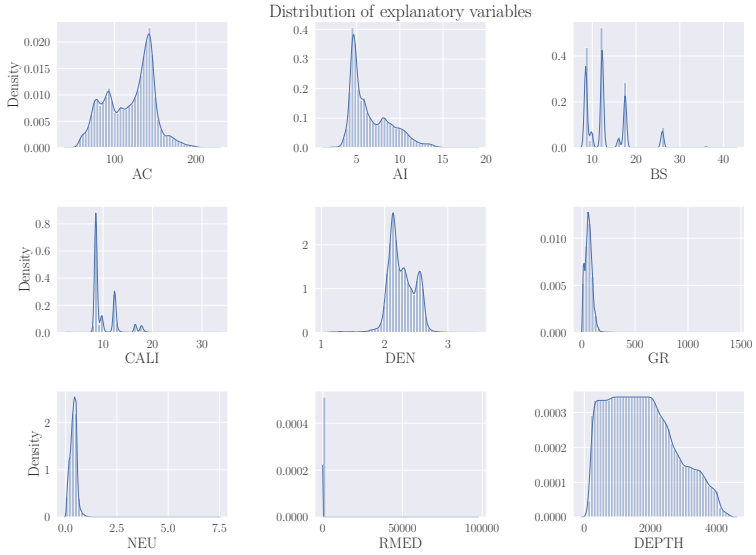


Figure 4.4: Distribution of the explanatory variables in the dataset. The variables are shown in the x-axis of each plot.

Again, different wells may give rise to widely different geophysical measurements, which likely results in a different set of distributions for the explanatory variables for each well. Figure 4.5 shows the distribution of the explanatory variables for a single well in the dataset. Observe how the distribution of a variable differs from the corresponding distribution over the full dataset. This can be explained by different geological formations resulting in widely different geophysical measurement across the wells. Moreover, this may indicate varying measurement noise and different calibration of the instruments across the wells.

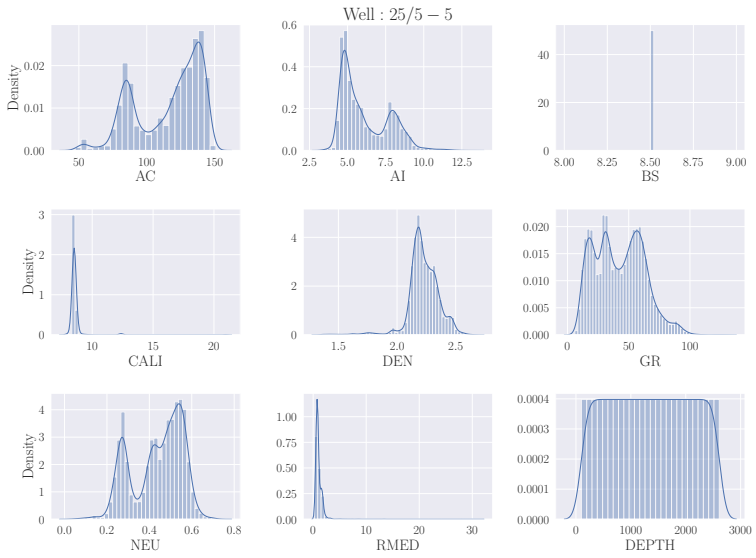


Figure 4.5: Distribution of the explanatory variables for a single well in the dataset. The variables are shown on the x-axis of the of each plot.

The dataset consists of 689 727 samples in total. However, a lot of the samples will be removed during preprocessing. The different preprocessing steps are described and justified in the next section.

4.2.1 Preprocessing and Feature Engineering

An integral part of any machine learning application is preprocessing the data before feeding it to the machine learning algorithm for training and testing.
The goal of preprocessing is to map the data into a form that is best suited for the training algorithm and neural network architecture and typically to reduce the dimensionality of the problem [5].

The first step in the preprocessing is to dispose of imperfect or missing measurements for the target variable, i.e. ACS. The dataset is equipped with a binary variable BADACS that represents whether the measurement of the acoustic log for S-waves is good or bad. These are measurements flagged by petrophysicists as being flawed, erroneous or in general not trustworthy. Having this flag makes it trivial to filter out imperfect measurements. The number of samples in the dataset is reduced from 689 727 to 256 014 samples after filtering out poor and missing measurements.

The next step is to select the explanatory variables. Initially, the raw dataset consists of 44 different logs or features for each sample. The variables in Table 4.1 are selected to reduce the dimensionality and ease interpretation of the models. The set of variables is chosen based on the same variables used in manual interpretation of the data.

A simple mean-imputer is applied to deal with missing values, which sets the missing values for a feature to be the mean value of the feature over the dataset. There exist other, more sophisticated imputing methods. However, the mean-imputer is chosen as it previously was shown to yield good results on statistical models deployed by Aker BP using the same dataset.

The final step of the preprocessing is to scale the features to a common scale. This is done well-wise, such that each variable is scaled to have zero mean and unit variance for each well. The well-wise standardization is chosen because different wells may have instruments that are calibrated differently for the variables in the dataset. Moreover, the different wells are located in potentially widely different geological formations, causing the variables to have different distributions across the wells. All variables except depth are standardized. This is because depth as a feature does not make sense unless the values are in absolute terms. In particular, if there are systematic differences in depths between wells, we lose valuable information about the depth when standardizing in a well-wise manner. Another option is to scale depth globally over the entire training set, but we choose to leave the depth unscaled.

It is important to note that the well-wise scaling of the target variable restricts the analysis to the standardized space. This is because we are using the values of the target variable to standardize, which is unavailable when making predictions. We cannot return to the original scale without using the information of the target variable. However, this detail will not influence the analysis other than being restricted to the standardized space. We will come back to this choice in the discussion in Chapter 6.

Feature engineering is a process where the goal is to incorporate domain knowledge and extract more information from the selected features in the dataset. Applying feature engineering is a common practice in many machine learning applications. The following features are engineered from the selected features for both datasets, except for the target variable and depth.

i. Gradients

ii. Rolling window functions for the:

   i) Mean value

   ii) Minimum value

   iii) Maximum value

Each selected feature, except for the target variable and depth, will have four additional features in the final, preprocessed dataset. The set of engineered features are included to exploit non-local information in the modelling, i.e. information outside the point of measurement, for all the instances in the dataset. The preprocessed dataset contains 42 explantory variables after including the engineered features.

The gradient of a feature indicates how the value of the feature is changing at all points of measurements, and is approximated using second-order central differences in the interior, and first-order forward- and backward differences at the boundaries[1]. The gradient of a feature $\{x_i\}_{i=1}^N$ is estimated as follows

$$
\begin{aligned}
x_i^{(gradient)} &\approx \frac{x_{i+1} - x_{i-1}}{2h},\ i = 2, \ldots, N-1, \\
x_{i=1}^{(gradient)} &\approx \frac{x_{i+1} - x_i}{h}, \\
x_{i=N}^{(gradient)} &\approx \frac{x_i - x_{i-1}}{h},
\end{aligned}
$$

where the feature is assumed to be evenly spaced with distance h between measurements. Because the measurements are taken at every depth, and depth is uniformly distributed within each well (see Figure 4.5), this assumption is perfectly valid for our data.

The rolling window features are calculated using a window size of 4 samples, meaning that the new features represents the mean, minimum and maximum values over 4 samples for each sample in the dataset. The rolling window functions for feature $\{x_i\}_{i=1}^N$ are computed as follows.

$$
x_i^{(mean)} = \begin{cases} \frac{1}{4}\sum_{j=0}^{4-1} x_{i+j},\ i = 1, \ldots, N-(4-1) \\ \frac{1}{4}\sum_{j=0}^{4-1} x_{i-j},\ i = N-2, \ldots, N \end{cases}
$$

$$
x_i^{(max)} = \begin{cases} \max\{x_{i+j}\}_{j=0}^{4-1},\ i = 1, \ldots, N-(4-1) \\ \max\{x_{i-j}\}_{j=0}^{4-1},\ i = N-2, \ldots, N \end{cases}
$$

$$
x_i^{(min)} = \begin{cases} \min\{x_{i+j}\}_{j=0}^{4-1},\ i = 1, \ldots, N-(4-1) \\ \min\{x_{i-j}\}_{j=0}^{4-1},\ i = N-2, \ldots, N \end{cases}
$$

[1]Forward difference for the first sample, and backward difference for the last sample in the dataset.

4.2.2 Train/Test split

The preprocessed dataset is split into a training and a test set to measure the generalization capabilities of the models. The models are trained using the training set, and performance is measured on the test set. The test set should be kept away from the models until performance is measured, as the performance on the test set serves as a proxy for how well the models generalize to unseen data.

The training set is constructed by randomly sampling a subset of the total number of wells in the dataset, setting aside the remaining wells for the test set. The fraction of training wells is chosen to be 75% of the total number of wells. The training set is further split into a training and validation set to estimate the test error during training. The validation set is constructed similarly to the test set and consists of a random subset of 10% of the wells in the training set.

The dataset is split in a well-wise manner to avoid data leakage. Because the standardization of features in the preprocessing scheme is done over each well, we avoid flow of information between the training and test set by splitting the data in a well-wise manner. Moreover, it makes sense to split the data in this fashion due to different distributions of both target and explanatory variables across the wells (see Figure 4.3-4.5).

In Table 4.2 we see the number of samples and wells over the different datasets.

Table 4.2: Number of samples and wells in the training-, validation- and test set, as well as the fraction of the entire dataset.

	Training set	Validation set	Test set	Total
Samples	185 153	17 560	53 301	256 014
Wells	24	2	8	34
Fraction	67.5 %	7.5 %	25 %	100 %

In Figure 4.6 we see the distribution of the target variable across the training, validation and test set. Note that the below distributions are over the full datasets, i.e. independent of the wells within each dataset. Figures of the well-wise distribution of the target variable across the different datasets are found in Appendix A.

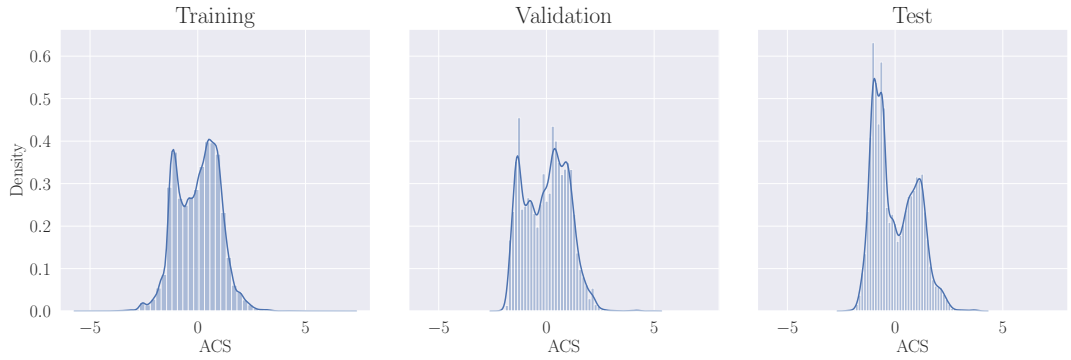


Figure 4.6: Distribution of the target variable across the training, validation and test set.

From the above figure we oberve that the dataset are distributed equally, althought there are some differences between the training and test set.

4.3 Preliminary analysis

A preliminary analysis is carried out on the dataset to justify the model choice. The analysis is performed using a random subset containing 10 000 samples of the preprocessed dataset.

The preliminary analysis consists of fitting a linear model using the regression dataset and constructing diagnostic plots to look into the model assumptions (see section 2.2). The results of the preliminary analysis will be used to justify the model choice for the regression model.

Figure 4.7 shows the residual plot using the linear model, and is a plot of the residuals $\hat{\boldsymbol{\varepsilon}}$ against the fitted values $\hat{\boldsymbol{y}}$. The residuals are estimates of the error term in the linear model, and shows the amount of variability in the data that the model cannot explain.

Figure 4.7: Residual plot for the linear model, where the raw residuals are plotted against the fitted values provided by the model. The red curve is a local regression line fit to the residuals to detect patterns.

It is clear that there is a non-linear relationship between the residuals and the fitted values, shown by the red LOWESS[2] curve. This suggests that a non-linear model is better suited for explaining the variability in the dataset. One can also see that the residuals have a non-constant variance over the range of fitted values. However, as the raw residuals are inherently heteroscedastic (see (2.5)), one cannot use the residual plot to assert whether the noise is heteroscedastic.

The raw residuals are standardized to yield a constant variance to fix the problem of inherent heteroscedasticity. The scale-location plot in Figure 4.8 is a plot of the square root of the standardized residuals against the fitted values. If the homoscedasticity assumption of the linear model is met, the standardized residuals should show a constant variance.

We observe that the variance of the standardized residuals is varying over the range of fitted values. Thus, the homoscedasticity assumption of the linear model is not met, suggesting using a heteroscedastic model for the error term.

The empirical quantiles of the standardized residuals are plotted against the theoretical quantiles of the standard normal distribution to check for normality of the error term in the linear model. The resulting Q-Q plot, as well as the distribution

[2]A local regression line fit to the residuals

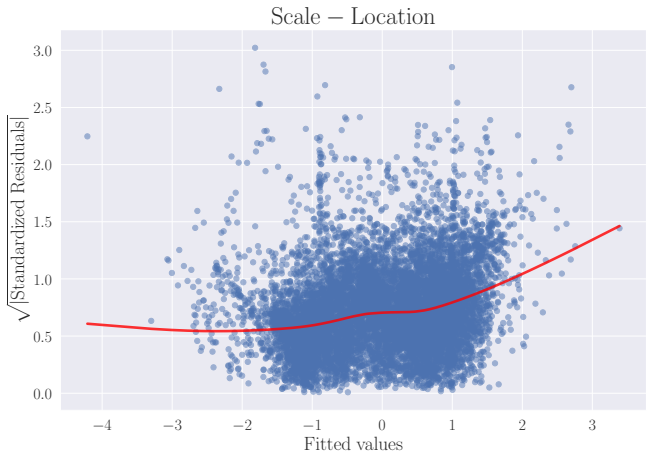
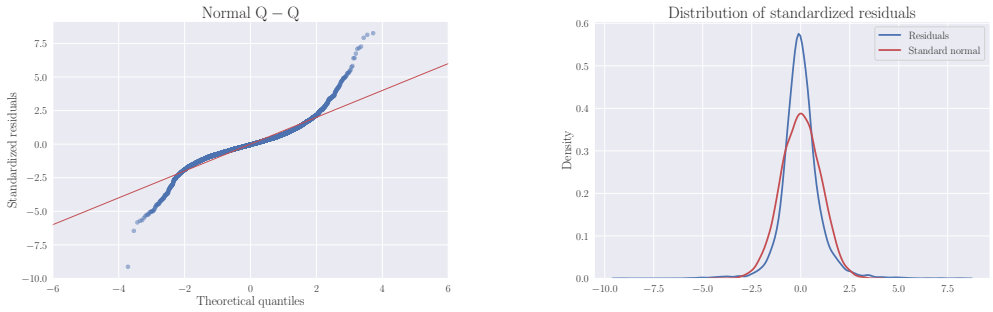


Figure 4.8: Scale-location plot a the linear, homoscedatic model, where the square root of the standardized residuals are plotted against the fitted values provided by the model. The red curve is a local regression line fit to the standardized residuals to detect patterns.

of the standardized residuals and the standard normal distribution, are shown in Figure 4.9.



(a) Empirical quantiles for the standardized residuals (blue) and theoretical quantiles of the standard normal distribution (red)

(b) Standardized residuals (blue) and the standard normal distribution (red)

Figure 4.9: (a) Normal Q-Q plot and (b) distribution of the standardized residuals and the standard normal distribution.

From Figure 4.9 we observe that the standardized residuals follow the standard normal distribution in the interior of the domain while having slightly heavier tails. This means that we observe more extreme values of the standardized residuals than expected from a standard normal distribution. Since the residuals capture the discrepancy between the true and estimated values for the target variable, this behaviour might indicate that the linear model does not perform sufficiently well.

Based on the preliminary analysis, we will use a non-linear model for the regression problem, with a normally distributed error term with heteroscedastic variance. Additionally, we will consider a homoscedsatic model and compare performance. The following section will present and describe the models in more detail.

4.4 Models and Architectures

4.4.1 Mathematical model

The below model aims to model the acoustic log for shear waves y_i given a set of well-log measurements $\boldsymbol{x}_i$ for sample $i = 1, ..., N$ in the dataset. The relationship between the explanatory variables $\boldsymbol{x}_i$ and the target variable y_i is assumed to be non-linear and modelled by a feedforward neural network $f(\boldsymbol{x}_i, \phi)$. Furthermore, a heteroscedastic noise model $g(\boldsymbol{x}_i, \psi)$ is assumed, meaning that each sample is assumed to be associated with an individual level of noise. These two assumptions are drawn from the diagnostic plots in Section 4.3 above.

We consider the following mathematical model for the acoustic log.

$$\left.\begin{aligned} &y_i = z_i + \varepsilon_i, \\ &z_i = f(\boldsymbol{x_i}, \boldsymbol{\phi}), \\ &\varepsilon_i \sim \mathcal{N}(0, s_i^2), \\ &s_i = g(x_i, \boldsymbol{\psi}),\ i = 1, \ldots, N, \\ &\phi_j \sim \mathcal{N}(a_j, b_j^2),\ j = 1, \ldots, K_\phi, \\ &\psi_k \sim \mathcal{N}(c_k, d_k^2),\ k = 1, \ldots, K_\psi. \end{aligned}\right\} \in \mathbb{R}, \tag{4.1}$$

where y_i is a measurement of the true, unobserved acoustic S-wave log z_i, subject to additive noise ε_i. The latent variables $\boldsymbol{\phi}, \boldsymbol{\psi}$ represents the parameters of the prediction and noise models, respectively.

For a fully Bayesian treatment, we need to specify prior distributions for both sets of parameters. The distribution of the parameters is then updated using VI during training, approximating the corresponding posterior distributions.

Given $\boldsymbol{\phi}, \boldsymbol{\psi}$ and a set of explanatory variables $\boldsymbol{x}^*$, the conditional acoustic log for the S-wave is calculated as $z = f(\boldsymbol{x}^*, \phi)$ and the measurement is represented by the following generative model.

$$\begin{aligned} &y|\boldsymbol{x}^*, \boldsymbol{\phi}, \boldsymbol{\psi} \sim \mathcal{N}(f(\boldsymbol{x}^*, \boldsymbol{\phi}), g(\boldsymbol{x}^*, \boldsymbol{\psi})^2) \in \mathbb{R}, \\ &\boldsymbol{x}^* \in \mathbb{R}^p,\ \boldsymbol{\phi} \in \mathbb{R}^{L_{\text{pred}}+1},\ \boldsymbol{\psi} \in \mathbb{R}^{L_{\text{noise}}+1}, \end{aligned} \tag{4.2}$$

where p is the number of explanatory variables and $L_{\{\text{pred, noise}\}}$ the number of hidden layers of the predictive network and the noise model, respectively.

The measurements of the acoustic S-wave log is subject to epistemic uncertainty through the parameters of the prediction model $f(\boldsymbol{x}, \psi)$ and aleatoric uncertainty through the noise model $g(\boldsymbol{x}, \psi)$, aiming to capture measurement error and the inherent randomness of the data-generating process.

The model in (4.1) is illustrated graphically as a probabilistic graphical model (PGM) using plate notation in Figure 4.10. Each plate represents repeated variables in the model, and the number of repetitions is marked on each plate. Random variables, i.e. variables associated with a probability distribution, are inscribed by circles, and the observed random variable is marked with a gray background. Dependencies between variables are marked with arrows. The dependency $x_i \rightarrow \varepsilon_i$ shows that the noise model is input dependent, i.e. heteroscedastic.

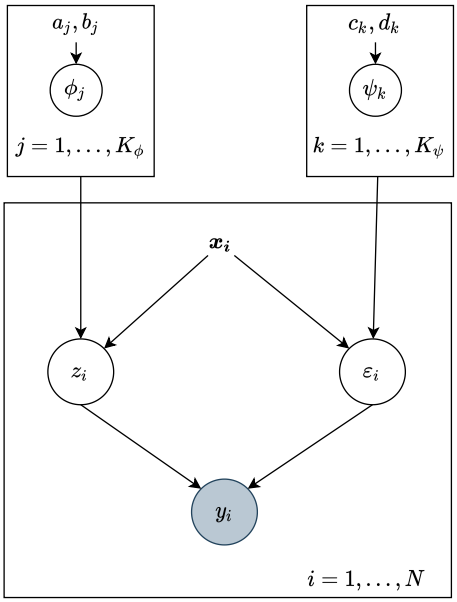


Figure 4.10: Probabilistic graphical model for the problem. Random variables are inscribed with a circle. The observed random variable is marked with a gray background.

The initial belief in the parameters of the models is encoded into the prior distributions. We assume that the priors are fully factorized normal distributions, meaning that the joint distribution of the parameters of a model can be expressed as a product of the marginal distribution of all the parameters in the model[3]. Thus, the prior distributions for the parameters in the prediction and noise model take the following form.

$$p(\boldsymbol{\phi}) = \prod_{j=1}^{K_\phi} \mathcal{N}(\phi_j | a_j, b_j^2),$$
$$p(\boldsymbol{\psi}) = \prod_{k=1}^{K_\psi} \mathcal{N}(\psi_k | c_k, d_k^2).$$

Further on, the latent variables $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are collected into the single latent variable $\boldsymbol{\theta}$. This allows to simplify the notation for the prior distribution of the latent variables, by further assuming independence between the parameters of the prediction and noise model.

$$\begin{aligned} p(\boldsymbol{\theta}) = p(\boldsymbol{\phi})p(\boldsymbol{\psi}) &= \prod_{i=1}^{K} \mathcal{N}(\theta_i | \bar{\mu_i}, \bar{\sigma_i}^2), \\ \bar{\boldsymbol{\mu}} = (\bar{\mu_1}, \ldots, \bar{\mu_K}) &= (a_1, \ldots, a_{K_\phi}, c_1, \ldots, c_{K_\psi}) \in \mathbb{R}^K, \\ \bar{\boldsymbol{\sigma}} = (\bar{\sigma_1}, \ldots, \bar{\sigma_K}) &= (b_1, \ldots, b_{K_\phi}, d_1, \ldots, d_{K_\psi}) \in \mathbb{R}^K, \end{aligned} \tag{4.3}$$

where $K = K_\phi + K_\psi$ is the total number of latent variables in the prediction and noise models. The distribution of each latent variable is parametrized by a mean and standard deviation, such that the total number of parameters is $2K$.

The distribution of the latent variables is updated during training using variational inference, approximating the true posterior distribution of the latent

[3]By assuming independence between the parameters

variables by a variational distribution. We have seen that in this case, the inference problem reduce down to an optimization problem that yields the variational distribution in the family $\mathcal{Q}$ that is most similar to the true posterior.

$$\begin{aligned} q_\lambda(\boldsymbol{\theta})^* &= \underset{q_\lambda \in \mathcal{Q}}{\operatorname{argmin}} D_{KL}[q_\lambda(\boldsymbol{\theta}) || p(\theta|\mathcal{D})] \\ &= \underset{q_\lambda \in \mathcal{Q}}{\operatorname{argmax}} \mathcal{L}(q_\lambda(\boldsymbol{\theta})) \\ &= \underset{q_\lambda \in \mathcal{Q}}{\operatorname{argmin}} -\mathcal{L}(q_\lambda(\boldsymbol{\theta})), \end{aligned}$$

where $\mathcal{L}(q_\lambda(\boldsymbol{\theta}))$ is the ELBO loss

$$\mathcal{L}(q_\lambda(\boldsymbol{\theta})) = \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] - D_{KL}[q_\lambda(\boldsymbol{\theta})||p(\theta)]. \tag{4.4}$$

In the case of the above model assumptions in (4.1), (4.2) and (4.3), the ELBO loss can be derived analytically. The derivation is found in Appendix B. Following the derivation we end up with the following expressions for the terms in the ELBO loss.

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = -\frac{N}{2}\log 2\pi - \sum_{i=1}^{N}\left[\log g(\boldsymbol{x_i}, \boldsymbol{\psi}) - \frac{1}{2}\left(\frac{y_i - f(\boldsymbol{x_i}, \boldsymbol{\phi})}{g(\boldsymbol{x_i}, \boldsymbol{\psi})}\right)^2\right],$$

$$D_{KL}[q_\lambda(\boldsymbol{\theta})||p(\boldsymbol{\theta})] = \frac{1}{2}\sum_{i=1}^{K}\left[2\log\frac{\bar{\sigma}_i}{\sigma_i} + \left(\frac{\sigma_i}{\bar{\sigma}_i}\right)^2 + \left(\frac{\mu_i - \bar{\mu}_i}{\bar{\sigma}_i}\right)^2 - 1\right],$$

where the expected log-likelihood is estimated using Monte Carlo sampling.

$$\mathbb{E}_{q_\lambda(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx \frac{1}{B}\sum_{i=1}^{B}\log p(\mathcal{D}|\theta_i),$$

where θ_i denotes the realization of the ith Monte Carlo sample from the variational distribution for $i = 1, \ldots, B$ [7].

A common challenge when implementing Bayesian neural networks is setting the prior distribution of its parameters. By imposing a prior distribution for a parameter of interest, we are trying to encapsulate prior beliefs about the parameter using a probability distribution. For black-box models like deep neural networks, where the parameters offer no physical interpretation, the prior specification becomes non-trivial. As mentioned in Section 2.5, specifying meaningful priors for the parameters of deep neural networks is an active field of research, and for computational simplicity we will make use of a standard normal prior for all the parameters in the model. We are thus deploying the following prior,

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{K}\mathcal{N}(\theta_i|\bar{\mu_i} = 0, \bar{\sigma_i}^2 = 1).$$

By using the above prior distribution for the parameters of the model, we obtain the following simplified expression for the KL-divergence term in the ELBO-loss (4.4).

$$D_{KL}[q_\lambda(\boldsymbol{\theta})||p(\boldsymbol{\theta})] = \frac{1}{2}\sum_{i=1}^{K}\left[2\log\frac{1}{\sigma_i} + \sigma_i^2 + \mu_i^2 - 1\right],$$

where μ_i and σ_i are the parameters of the variational distribution for the ith parameter of the model.

In the next part of this section, we will describe the architecture of the different neural networks we consider.

4.4.2 Neural Network Architecture

The Bayesian neural network based on the two methods presented in Chapter 3 are implemented in a fully connected, feedforward manner. Both methods are implemented using the same structure and configuration of hyper-parameters to be able to compare performance appropriately. Furthermore, the predictive performance of the Bayesian neural networks is compared to a deterministic neural network with the same architecture.

We refer to the models implemented with Stochastic Gradient Variational Bayes as SGVB models. The models implemented with Monte Carlo Dropout are referred to as MC Dropout models. Both methods are implemented to model the aleatoric uncertainty in a heteroscedastic and homoscedastic manner, and we are thus concerned with four different model formulations. The models are implemented using three fully connected layers, each consisting of 100 computational neurons. Furthermore, the dataset consists of a single target variable with 42 explanatory variables. The number of input- and output nodes are thus 42 and 1, respectively.

The models are built in a modular fashion. Each intermediate module consists of a fully connected layer followed by a batch-normalization layer, a ReLU activation function (2.6), and a dropout layer with a 10% dropout rate. The module corresponding to the output layer consists of a fully connected layer followed by a linear activation function (2.7).

A computational graph representing the model architecture, as well as the different modules, are seen in Figure 4.11 below. The figure shows a homoscedastic model where the aleatoric uncertainty is assumed constant and a heteroscedastic model where the aleatoric uncertainty varies across samples in the dataset.

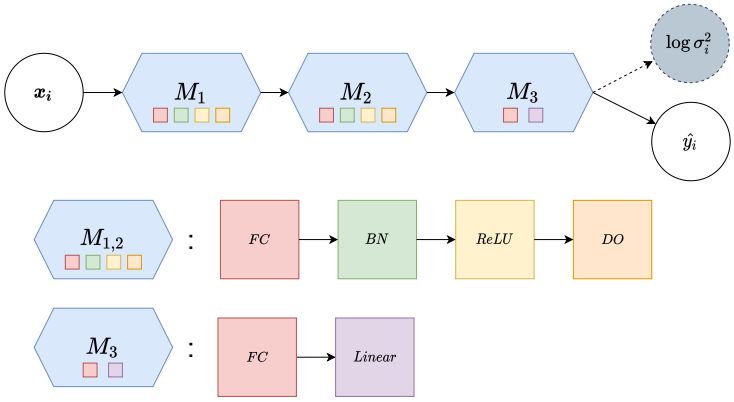


Figure 4.11: Architecture of the neural network models (top) for an input instance $\boldsymbol{x_i}$, with modules M_{1-3} (bottom). The intermediate modules $M_{1,2}$ consist of a fully connected (FC) layer followed by a batch-normalization (BN) layer, ReLU activation and a dropout (DO) layer. The output module M_3 consist of a fully connected layer followed by a linear activation function. In the homoscedastic setting (white output node) the output of the model is $\hat{y}_i$, while in the heteroscedastic setting (both output nodes) the outputs are $\hat{y}_i$ and $\log \sigma_i^2$.

The aleatoric uncertainty is treated as a model parameter optimized alongside the weights and biases by minimizing the loss for the homoscedastic models. In the heteroscedastic setting, the models are made multi-headed, where the output of the network for a single input instance is the predicted target variable as well as the estimated aleatoric uncertainty (see Figure 4.11).

The models are trained using the Adam optimizer with an initial learning rate of 10^{-4}, minimizing the respective loss functions. Each model formulation is concerned with a separate loss function depending on the method for implementing the BNN and the nature of the aleatoric uncertainty. The SGVB models we are trained using the ELBO loss (4.4), while the MC Dropout and deterministic models are trained using the negative log-likelihood loss (3.5)-(3.4). Consequently, we obtain a homoscedastic and heteroscedastic model for the deterministic architecture. Note that by training the deterministic models with the NLL loss, we obtain a probabilistic interpretation in terms of the aleatoric uncertainty. However, we will not leverage the aleatoric uncertainty in the deterministic models. The choice of loss function for the deterministic models is made to perform an appropriate comparison with the probabilistic models. The difference between the deterministic and probabilistic models is that the probabilistic models can reason about the uncertainty in the parameters of the models and the uncertainty in the data. In contrast, deterministic models are not able to do so.

To make sure the MC Dropout formulation is mathematically equivalent to performing variational inference on the neural network parameters, we need to train the models by adding a specific weight-decay parameter λ to the loss function. The weight-decay parameter is calculated using the additional hyper-parameters introduced by the prior specification of the MC Dropout models (see Table 3.1). The models are implemented using a prior length scale $l = 1.00$, corresponding to a standard normal prior distribution for the parameters of the network. The model precision is set to $\tau = 1.00$, which correspond to the following weight decay parameter for the parameters in the dropout layers.

$$\lambda_{Dropout} = \frac{l^2(1-p)}{2N\tau} = \frac{1.0^2(1-0.10)}{2 \cdot N \cdot 1.0} = 2.43 \cdot 10^{-6},$$

where $N = 185\ 153$ is the number of samples in the training set and $p = 0.10$ the dropout rate. Because dropout is only applied to the non-output layers of the network, the weight-decay parameter for the parameters in the output layer is slightly different. By not applying dropout we are setting the dropout rate to $p = 0.00$, and the resulting weight-decay parameter is

$$\lambda_{Output} = \frac{l^2(1-p)}{2N\tau} = \frac{1.0^2(1-0.00)}{2 \cdot N \cdot 1.0} = 2.70 \cdot 10^{-6}.$$

During training, we perform mini-batch optimization in batches of 100 samples. With 185 153 samples in the training set, we need to perform 1852 iterations every epoch. The models are trained for ten epochs, which corresponds to 18 520 iterations for optimizing the loss function and training each model.

The parameters of the MC Dropout models are initialized using the He-initialization [24] to avoid the problem of exploding and vanishing gradients. The aleatoric variance parameter in the homoscedastic models is initialized by drawing samples from a standard normal distribution. In the heteroscedastic models, the aleatoric variance is predicted using the multi-headed network, which can be seen as an auxiliary model. The parameters of the auxiliary model are

initialized with the He-initialization, similar to the models providing the predictions. For the SGVB models, the variational parameters representing the mean and standard deviation of the parameters in the network are initialized with a zero-mean normal distribution with standard deviation $\sigma = 0.001$. In both methods, all the biases are initialized to zero[4]. The prior distribution for the parameters are explicitly specified as a standard normal distribution for the SGVB method, while implicitly specified as a standard normal by the weight-decay parameter in the MC Dropout method.

The complete training configuration is summarized in Table 4.3 below.

Table 4.3: Hyper-parameters and training configuration for the neural networks.

Hyperparameter	**Value**
Input nodes	43
Output nodes	$1 - 2$
Hidden layers	3
Nodes in hidden layers	100
Non-output activation	ReLU
Output activation	Linear
Dropout rate	10%
Loss	ELBO, NLL
Optimizer	ADAM
Learning rate	10^{-4}
Prior length scale	1.0
Precision	1.0
Weight-decay (non-output)	$2.43 \cdot 10^{-6}$
Weight-deacy (output)	$2.70 \cdot 10^{-6}$
Batch size	100
Epochs	10
Iterations	18 520

To analyze the convergence properties of the epistemic uncertainty, i.e. the approximation uncertainty, we fit the above models using different fractions of the training set. Referring back to section 3.5 this is done over the range of $10 - 100\%$ of the entire training set, meaning that we need to fit 36 additional models[5]. To separate the approximation uncertainty from the model uncertainty, we use the same architecture described above for all these additional models. Moreover, to look into the effect of the dropout rate on the epistemic uncertainty of the MC Dropout models, we train two additional sets of models with dropout rates $p = \{0.30, 0.50\}$ for all the fractions of the training set. This is done for all the considered models to be able to compare the results appropriately.

We fit four models with varying model complexity for the homoscedastic and heteroscedastic SGVB models to look into the model uncertainty. One of the model complexities is the one described above. This accounts for three additional models for each of the two model formulations, resulting in six additional models. The MC Dropout method is omitted because it is not possible to fit a linear model with the method (see Section 3.1). The most simple model we consider is a Bayesian linear regression model with no hidden layers. Additionally, we consider a Bayesian

[4]Also the mean and standard deviation of the biases in the SGVB method

[5]We need to consider nine additional models for each of the four model formulations we consider

neural network with a single hidden layer. For the single-layer model, the hidden layer is equipped with batch-normalization, ReLU activation and a dropout layer, identical to the modules $M_{1,2}$ in Figure 4.11. The most complex model we consider is a Bayesian neural network with ten hidden layers, each being identical to the above-described modules with batch-normalization, ReLU activation and a dropout layer. The output layer for these models is identical to the output module M_3 in Figure 4.11, with a fully connected layer coupled with a linear activation function. To separate the model uncertainty from the approximation uncertainty, we are training all the different model complexities using the entire training set.

4.5 Deep Learning Framework

All considered models are implemented in the Python programming language using the PyTorch framework for deep learning [46]. Due to its simplicity, the MC Dropout method is implemented in the standard PyTorch framework without further extensions. However, a suitable configuration needs to be taken into account to obtain mathematical equivalence to the Bayesian Deep Gaussian Process. To implement the SGVB method, the deterministic feedforward layer of a neural network is extended to a probabilistic Bayesian layer using variational inference. The implementation of the latter method is based on the original article introducing the method [7].

The source code for implementing the methods and performing the analyses can be accessed in the GitHub repository for the thesis[6].

[6]The repository is found here: `https://github.com/christianlehre/thesis`

Chapter 5

Results

The first two sections of this chapter present the results of the two different methods for constructing BNNs. The presented results are shown as prediction curves, where the predicted and true values of the acoustic log for shear waves are plotted against depth for a single well in the test set. The prediction curves for the remaining wells in the test set are found in Appendix C. Furthermore, the predictions are equipped with 95% credible intervals. Doing so allows one to reason about the uncertainty of the predictions provided by the models. The predictive performance of the models are reported, and the performance of the two methods are compared with a deterministic neural network with a similar architecture. The performance is measured in a well-wise manner and over the entire test set, independent of wells.

A qualitative analysis of the aleatoric uncertainty is presented in the subsequent section, where the explanatory variables are plotted alongside the aleatoric variance provided by the heteroscedastic models. This is done to investigate whether the estimated aleatoric uncertainty captures uncertainty in the data.

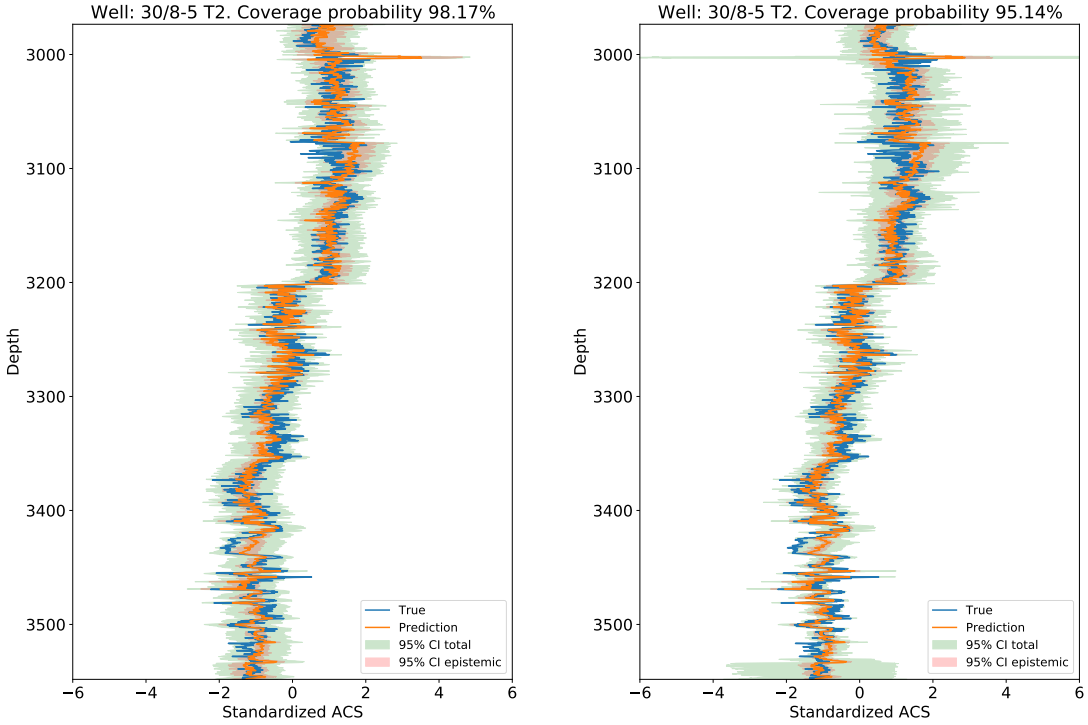
Next, the results regarding the quantitative evaluation of the uncertainty estimates are presented. The presented results are aggregated calibration curves across all the wells in the test set for each model. Separate calibration curves for all the wells in the test set are found in Appendix D for all the considered models.

In the final sections of this chapter, we present the results regarding the analysis of the different components of epistemic uncertainty.

It is important to note that the results are presented on a standardized scale, meaning that we present the predictions and true values of the response in terms of their corresponding z-scores (see Section 4.2.1). This is a result of the well-wise scaling of the response variable, which makes it impossible to return to the original scale without leveraging the true values of the response.

5.1 MC Dropout

The predictions curves for well 30/8-5 T2 using the MC Dropout models are shown in Figure 5.1. The homoscedastic model is in the left panel of the figure, while the heteroscedastic model is in the right panel. The predictions (orange), as well as the ground-truth values (blue), are plotted against depth, and the predictions are equipped with 95% credible intervals based on the total predictive uncertainty (green) and the epistemic uncertainty alone (red). The empirical coverage probability (3.13) is marked in the title of each plot and is based on the total predictive variance.



(a) Homoscedastic model

(b) Heteroscedastic model

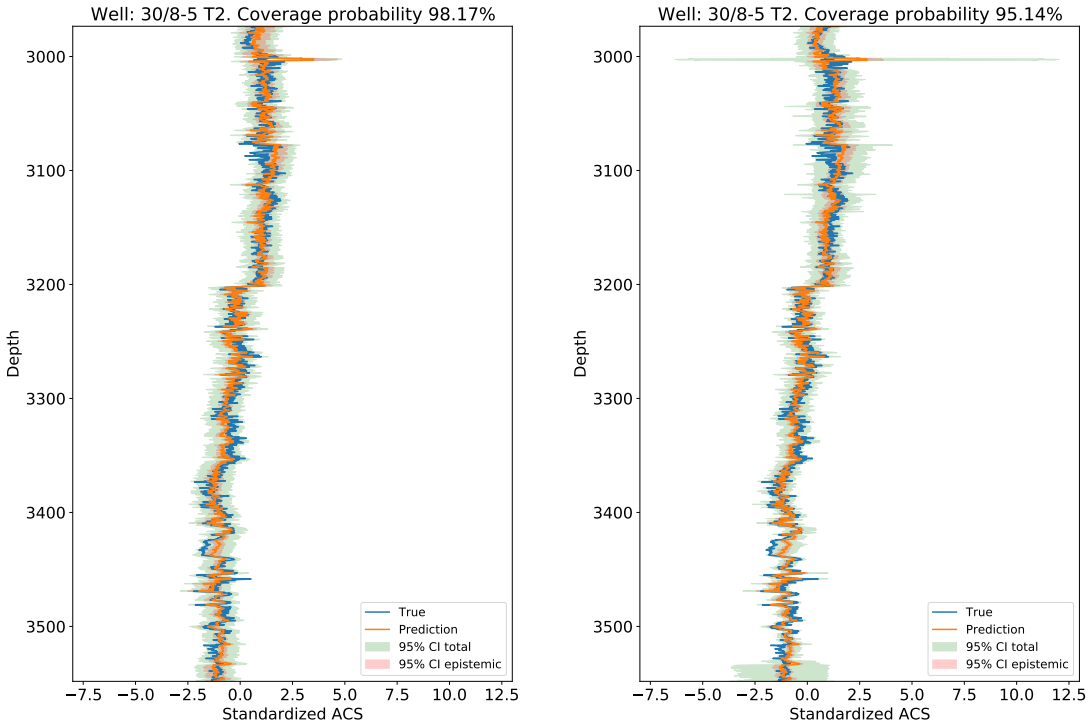
Figure 5.1: Predictions (orange) and corresponding 95% credible intervals for the total predictive uncertainty (green) and the epistemic uncertainty (red) for well 30/8-5 T2 in the test for the (a) homoscedastic and (b) heteroscedastic MC Dropout models. The true values are shown in blue, and the empirical coverage probability (3.13) is marked in the title of each plot.

We observe that the predictions from both models follow the true values well, although there are some discrepancies. As for the uncertainty estimates provided by the credible intervals, we see that there is very little epistemic uncertainty relative to the total predictive uncertainty. Hence, there is a lot of aleatoric uncertainty present in the data. For the homoscedastic model, i.e. where the aleatoric uncertainty is assumed constant, the empirical coverage probability is 98.17%. This is greater than what is expected by the 95% credible interval. Consequently, the provided credible interval is too wide, and the predictions are thus under-confident. For the heteroscedastic model, where the aleatoric uncertainty is allowed to vary among

samples, the empirical coverage is 95.14%, which is very close to the expected level of the credible interval.

There is a clear distinction between the models in the credible intervals provided by the total predictive uncertainty. However, the epistemic credible intervals are seemingly comparable. Consequently, we say that the differences in the total predictive credible intervals are attributed to the aleatoric uncertainty in the data. Hence, we see that there are intervals along the depth of the well where the heteroscedastic model estimates the aleatoric uncertainty to be greater than the homoscedastic model and other intervals where the aleatoric uncertainty is greater in the homoscedastic setting. In particular, for the heteroscedastic model, we observe a spike in the aleatoric uncertainty around 3000 m, and an interval of elevated aleatoric uncertainty from $\sim$ 3550 m to the end of the well. There is also an interval around 3040 $-$3150 m with frequently occurring spikes in the aleatoric uncertainty, albeit the levels are not as high as the former spike at $\sim$ 3000 m.

Although the heteroscedastic spike in aleatoric uncertainty goes outside the range of plotted values for the response in Figure 5.1, we added the figure to observe better the epistemic uncertainty given by the red credible intervals. Figure 5.2 shows the same predictions plot as in Figure 5.1, where the range of values for the response covers the entire spike in aleatoric uncertainty.



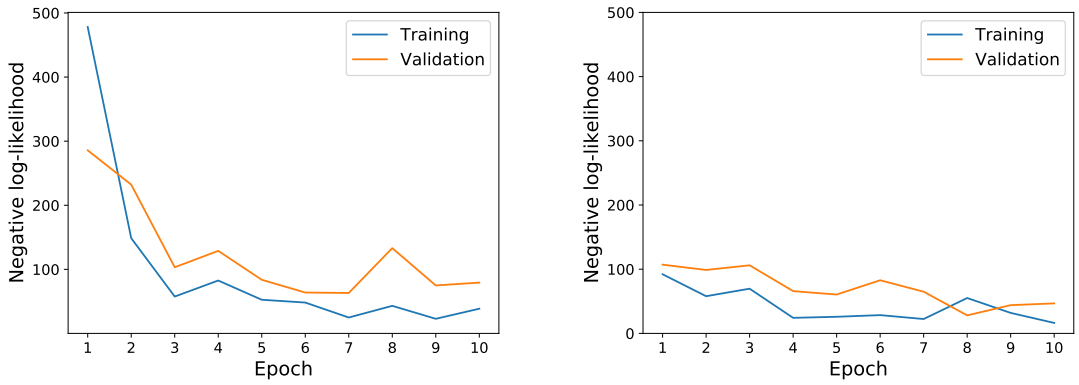
(a) Homoscedastic model

(b) Heteroscedastic model

Figure 5.2: Predictions (orange) and corresponding 95% credible intervals for the total predictive uncertainty (green) and the epistemic uncertainty (red) for well 30/8-5 T2 in the test for the (a) homoscedastic and (b) heteroscedastic MC Dropout models. The true values are shown in blue, and the empirical coverage probability (3.13) is marked in the title of each plot.

From Figure 5.2, it is clear that there is an anomaly in the aleatoric uncertainty around 3000 m for the heteroscedastic model and that the aleatoric uncertainty provided by the homoscedastic model is much lower. We will analyze the data and look for evidence of the increased levels of aleatoric uncertainty at particular depths in section 5.3.

The loss curves for the models are shown in Figure 5.3. The homoscedastic model is shown in the left panel, while the heteroscedastic model is shown in the right panel. The training and validation loss is marked in blue and orange, respectively.



(a) Homoscedastic model (b) Heteroscedastic model

Figure 5.3: Loss curves (3.5, 3.4) for the (a) homoscedastic and (b) heteroscedastic MC Dropout models. The training loss is marked in blue, while the validation loss is marked in orange.

We observe that both models converge properly from the above loss curves, and there is no sign of over or under-fitting. The most notable difference between the two models is the loss in the earlier epochs. After a single epoch, we see that the loss is much greater for the homoscedastic than the heteroscedastic model. However, the loss is decreasing rapidly, obtaining similar values to the heteroscedastic loss after three epochs. In the subsequent epochs, the loss decreases steadily towards convergence for both models.

To measure the predictive performance of the models, the MSE and MAE performance metrics (3.10, 3.11) are calculated. The calculations are performed in a well-wise manner as well as over the full test set. Doing so allows capturing the difference in predictive performance that is not possible to observe with the naked eye by looking at the prediction curves in Figures 5.1 and 5.2. The results of the well-wise predictive performance for the MC Dropout models are shown in Table 5.1, where the better performing model for each metrics is marked in bold. The row that corresponds to the well presented above is colored in yellow. The values are presented as credible intervals represented by a single standard deviation around the mean value of the distribution for each metric, obtained by the sampling approach described in Section 3.3.

We observe that the heteroscedastic model consistently outperforms the homoscedastic model, obtaining lower values for both performance metrics on all wells except one. This highlights the importance of modelling the aleatoric uncertainty in a heteroscedastic way for this problem. For well 25/4-10 S, the homoscedastic model performs better in terms of the MSE metric.

The total predictive performance of the models are reported in Table 5.2.

Table 5.1: Well-wise predictive performance metrics (3.10, 3.11) for the homoscedastic and heteroscedastic MC Dropout models. The better performing model is marked in bold for each well and metric, and the yellow row shows the above presented well.

	Homoscedastic		Heteroscedastic	
Well	MSE	MAE	MSE	MAE
30/11-7	0.10529 ± 0.00114	0.24866 ± 0.00093	**0.08559 ± 0.00075**	**0.21512 ± 0.00086**
25/4-10 S	**0.48009 ± 0.00782**	0.50581 ± 0.00461	0.49171 ± 0.00711	**0.49242 ± 0.00342**
30/11-9 ST2	0.12385 ± 0.00188	0.25485 ± 0.00177	**0.11505 ± 0.00168**	**0.25075 ± 0.00149**
30/6-26	0.13192 ± 0.00179	0.26842 ± 0.00184	**0.10342 ± 0.00148**	**0.24026 ± 0.00165**
30/11-10	0.10339 ± 0.00084	0.25423 ± 0.00098	**0.06603 ± 0.00065**	**0.19174 ± 0.00095**
30/8-5 T2	0.13395 ± 0.00267	0.27394 ± 0.00180	**0.12092 ± 0.00182**	**0.25919 ± 0.00160**
25/7-6	0.10195 ± 0.00317	0.24914 ± 0.00243	**0.08551 ± 0.00142**	**0.23276 ± 0.00221**
30/11-11 S	0.11320 ± 0.00132	0.25360 ± 0.00149	**0.11249 ± 0.00149**	**0.23722 ± 0.00137**

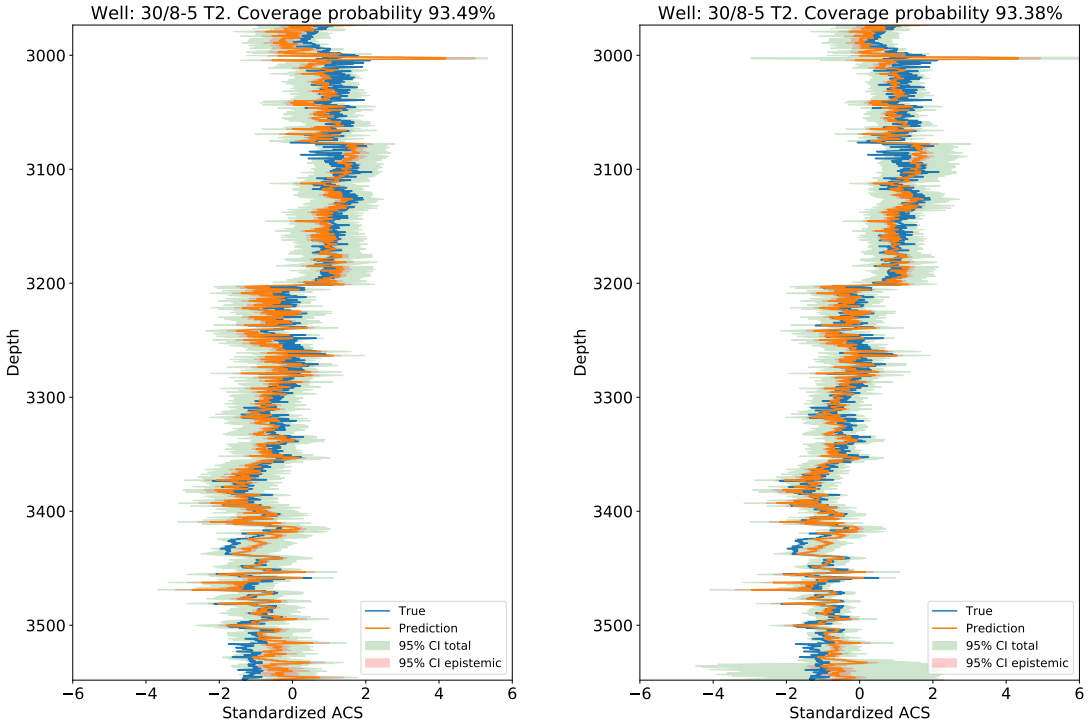
Table 5.2: Predictive performance over the full test set for the homoscedastic and heteroscedastic MC Dropout models. The better performing model is marked in bold for each metric.

	Homoscedastic	Heteroscedastic
MSE	0.10037 ± 0.00045	**0.09178 ± 0.00037**
MAE	0.23539 ± 0.00050	**0.21869 ± 0.00043**

We see that the heteroscedastic model outperforms the homoscedastic model in both performance metrics. This is expected by the consistently better performance seen by the heteroscedastic model in Table 5.1.

5.2 Stochastic Gradient Variational Bayes

The predictions curves for well 30/8-5 T2 using the SGVB models are shown in Figure 5.4. The homoscedastic model is in the left panel of the figure, while the heteroscedastic model is in the right panel. The predictions (orange), as well as the ground-truth values (blue), are plotted against depth, and the predictions are equipped with 95% credible intervals based on the total predictive uncertainty (green) and the epistemic uncertainty alone (red). The empirical coverage probability is marked in the title of each plot and is based on the total predictive variance.



(a) Homoscedastic model

(b) Heteroscedastic model

Figure 5.4: Predictions (orange) and corresponding 95% credible intervals for the total predictive uncertainty (green) and the epistemic uncertainty (red) for well 30/8-5 T2 in the test for the (a) homoscedastic and (b) heteroscedastic SGVB models. The true values are shown in blue, and the empirical coverage probability (3.13) is marked in the title of each plot.

Similar to the MC Dropout models, the predictions provided by the SGVB models follows the true values well, although there are some discrepancies. There are no clear distinctions between the predictions provided by the different methods, other than the discrepancy between the true and predicted values being slightly greater for the SGVB models in the outer extremities of the well, in both the homoscedastic and heteroscedastic setting.

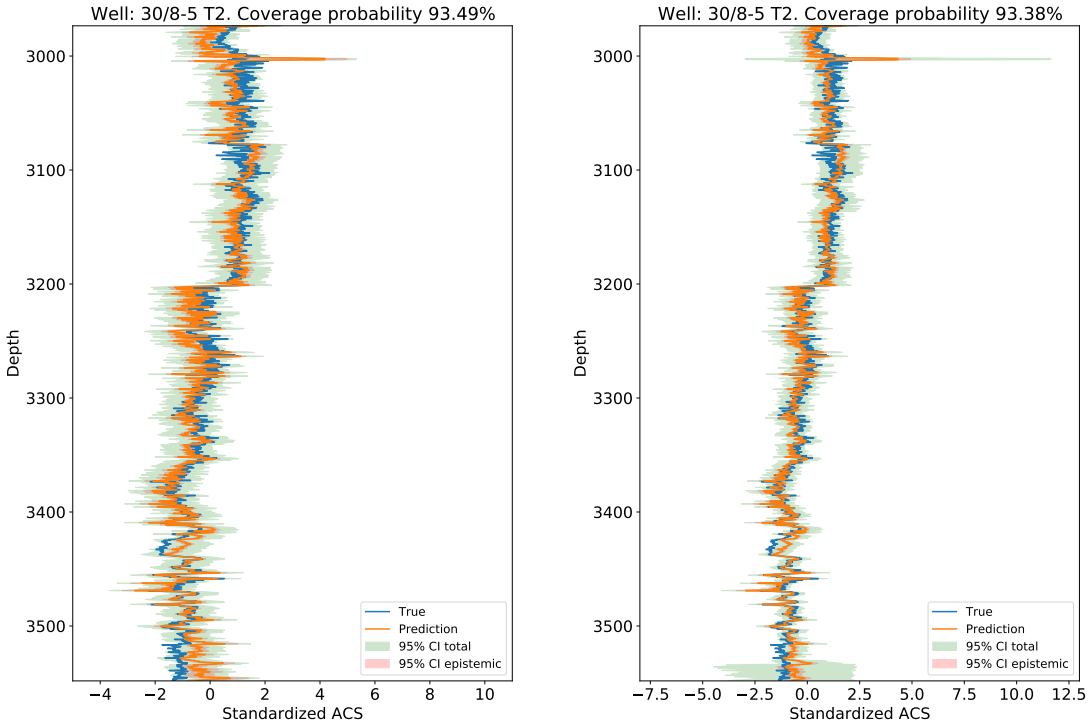
Regarding the uncertainty provided by the credible intervals, it appears that the epistemic uncertainty is practically negligible and even smaller compared to the MC Dropout models. Hence, the total predictive uncertainty is mainly attributed to

the aleatoric uncertainty in the data. Moreover, the total predictive uncertainty is seemingly comparable to the MC Dropout model.

The empirical coverage probabilities provided by the total predictive variance of the models predictions are slightly lower than what we can expect from the 95% probability level of the credible interval. Consequently, the corresponding credible intervals are too tight, and the predictions are thus over-confident. The homoscedastic model reports an empirical coverage probability of 93.49%, while the heteroscedastic model gives a 93.38% coverage probability.

Similar to the heteroscedastic MC Dropout model, the corresponding SGVB model has a spike in the aleatoric uncertainty at 3000 m depth, as well as an interval of elevated aleatoric uncertainty from $\sim$ 3700 m to the end of the well. Moreover, we observe that the credible interval is broader for the heteroscedastic SGVB model than the MC Dropout model presented in the previous section.

In Figure 5.4 the aleatoric uncertainty spike extends beyond the plotted range of values for the response. Figure 5.5 shows the same prediction plot, where the range of plotted values covers the entire spike.



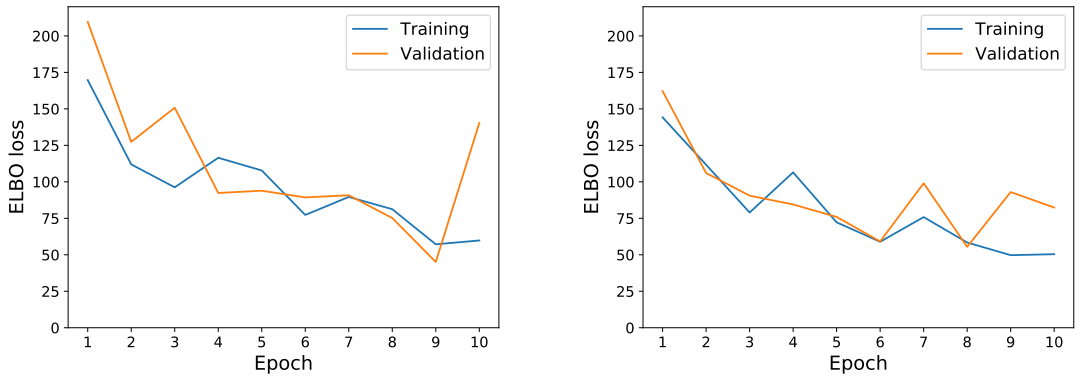
(a) Homoscedastic model

(b) Heteroscedastic model

Figure 5.5: Predictions (orange) and corresponding 95% credible intervals for the total predictive uncertainty (green) and the epistemic uncertainty (red) for well 30/8-5 T2 in the test for the (a) homoscedastic and (b) heteroscedastic SGVB models. The true values are shown in blue, and the empirical coverage probability (3.13) is marked in the title of each plot.

It is clear from Figure 5.5 that there is an anomaly in the aleatoric uncertainty at $\sim$ 3000 m. We will come back to this in section 5.3, where we will look for evidence of the aleatoric uncertainty in the data.

The loss curves for the models are shown in Figure 5.6. The left panel shows the homoscedastic model, and the right panel shows the heteroscedastic model. The training loss and validation loss is marked in blue and orange, respectively.



(a) Homoscedastic model (b) Heteroscedastic model

Figure 5.6: Loss curves (3.7) for the (a) homoscedastic and (b) heteroscedastic SGVB models. The training loss is marked in blue, while the validation loss is marked in orange.

Based on the above loss curve, it looks like neither of the models has fully converged, although the loss decreases steadily with the number of epochs. Moreover, we observe that the loss curves are pretty noisy. However, this may be an attribute of the stochastic optimization.

The well-wise predictive performance of the models are reported in Table 5.3, where the row corresponding to the presented well in Figures 5.4 and 5.5 is marked in yellow. The model that performs better in terms of each performance metric is marked in bold for each well. Similar to the MC Dropout models, the values are presented as credible intervals represented by a single standard deviation around the mean value of the distribution of each metric. We observe that the heteroscedastic SGVB model consistently outperforms the homoscedastic model in terms of both the performance metrics we consider. Moreover, by comparing the performance with the MC Dropout model in Table 5.2, we observe that the performance metrics for the SGVB models are bigger than for the MC Dropout models for all the wells. Consequently, the predictive performance is better for the MC Dropout models.

The predictive performance of the SGVB models over the full test set is reported in Table 5.4 below. Just as for the MC Dropout models, we see that the heteroscedastic SGVB model outperforms the homoscedastic model in terms of the considered performance metrics over the entire test set. This is consistent with the systematic better performance of the heteroscedastic model for all the wells shown in Table 5.4

We report the predictive performance over the entire test set for all the models we consider for a final comparison of predictive performance. In addition to the probabilistic MC Dropout and SGVB models, we report the predictive performance of a deterministic model with a similar architecture. The results are reported in Table 5.5, where the better performing model in terms of the performance metrics in each setting[1] is marked in bold.

[1]referring to the nature of the aleatoric uncertainty

Table 5.3: Well-wise predictive performance metrics (3.10, 3.11) for the homoscedastic and heteroscedastic SGVB models. The better performing model is marked in bold for each well and metric. The yellow row shows the well presented above.

	Homoscedastic		Heteroscedastic	
Well	MSE	MAE	MSE	MAE
30/11-7	0.15315 ± 0.03021	0.27989 ± 0.02660	**0.10337 ± 0.02499**	**0.23178 ± 0.02769**
25/4-10 S	0.73720 ± 0.06574	0.66689 ± 0.03626	**0.67566 ± 0.05298**	**0.62057 ± 0.02866**
30/11-9 ST2	0.20350 ± 0.03487	0.28119 ± 0.02062	**0.16109 ± 0.02662**	**0.27042 ± 0.01493**
30/6-26	0.18307 ± 0.01990	0.29051 ± 0.01433	**0.11108 ± 0.00783**	**0.24960 ± 0.00935**
30/11-10	0.21669 ± 0.05689	0.34469 ± 0.04265	**0.18715 ± 0.03902**	**0.32363 ± 0.03793**
30/8-5 T2	0.24164 ± 0.02318	0.36032 ± 0.01994	**0.16249 ± 0.01359**	**0.28349 ± 0.01272**
25/7-6	0.06590 ± 0.00950	0.20124 ± 0.01709	**0.05425 ± 0.00754**	**0.16853 ± 0.01178**
30/11-11 S	0.20429 ± 0.02365	0.32203 ± 0.01864	**0.17672 ± 0.02548**	**0.26638 ± 0.01548**

Table 5.4: Predictive performance over the full test set for the homoscedastic and heteroscedastic SGVB models. The better performing model is marked in bold for each metric.

	Homoscedastic	Heteroscedastic
MSE	0.11659 ± 0.01715	**0.09109 ± 0.00812**
MAE	0.24990 ± 0.02137	**0.21777 ± 0.01128**

Table 5.5: Summary of predictive performance over the full test set for all considered models. The best performing model in each setting, i.e. homoscedastic and heteroscedastic, is marked in bold for each metric.

	Homoscedastic		Heteroscedastic	
	MSE	MAE	MSE	MAE
MC Dropout	**0.10037 ± 0.00045**	**0.23539 ± 0.00050**	0.09178 ± 0.00037	0.21869 ± 0.00043
SGVB	0.11659 ± 0.01715	0.24990 ± 0.02137	**0.09109 ± 0.00812**	0.21777 ± 0.01128
Deterministic	0.11908	0.27258	0.09193	**0.20618**

For the Homoscedastic models, we see that the MC Dropout model outperform the SGVB- and deterministic model in both considered performance metrics. We observe that the SGVB model performs better than the other models in terms of the MSE metric for the heteroscedastic models. Furthermore, we see that the deterministic model outperforms both the probabilistic models in terms of the MAE metric.

5.3 Qualitative Analysis

This section will present a qualitative analysis of the aleatoric uncertainty of the heteroscedastic models for the well presented in the above sections. By plotting the aleatoric variance as a function of depth, alongside the explanatory variables, we can investigate whether the estimated aleatoric uncertainty seen in Figures 5.1 and 5.4 respond to potentially occurring noise in the data. The homoscedastic models are omitted from this analysis because the aleatoric uncertainty provided by such models is constant. We can thus not investigate how noise in the explanatory variables affects the aleatoric uncertainty at particular depths.

We have omitted the BS log as well as the rolling window features from the analysis. The BS log, i.e. diameter of the borehead, is omitted because it is a feature of the borehead rather than the rock formations in the subsurface. However, the CALI log is strongly correlated with BS, as it is a physical measurement of the diameter of the well. Omitting the rolling window features is done because they aim to capture non-local information about the features. The rationale behind omitting non-local information is that we want to look at how noise in the data translates to the aleatoric uncertainty at particular depths. However, we can incorporate the non-local information into the analysis simply by looking at the logs.

We will consider three different intervals of depth along the well, each associated with different attributes of the aleatoric uncertainty described in the above sections.

The aleatoric variance in the leftmost plot in the below figures is plotted against depth for the heteroscedastic MC Dropout model (green) and SGVB model (red). The remaining plots show a subset of the explanatory variables against depth, where the logs are plotted in blue and the corresponding gradients in orange.

In Figure 5.7 we consider an interval around the depths where we observe a distinct spike in the aleatoric uncertainty.

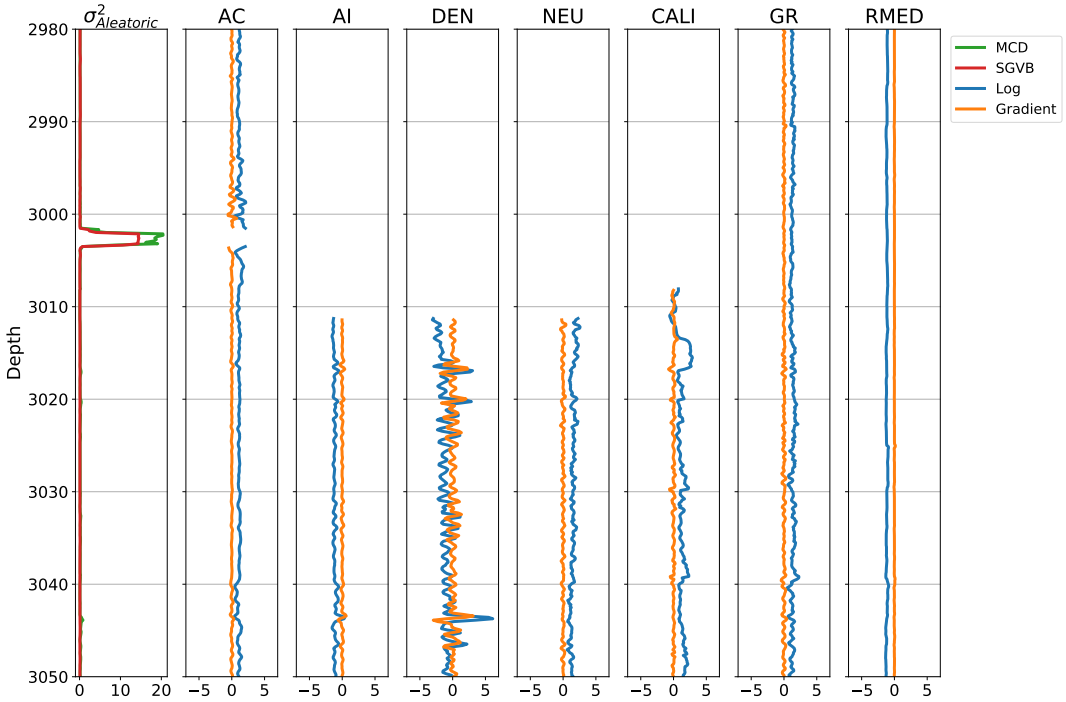


Figure 5.7: Aleatoric variance alongside a subset of the explanatory variables against depth for well 30/8-5 T2 around the spike in aleatoric uncertainty. In the leftmost plot, the aleatoric variance for the MC Dropout model is marked in green, while the SGVB model is marked in red. For the remaining figures, the logs are marked in blue, and the corresponding gradients in orange. The different logs are marked in the title of each plot.

We observe that both models behave similarly in terms of the provided aleatoric uncertainty, with a spike at $\sim 3000\,\text{m}$. Moreover, we observe that the MC Dropout is more affected, as the relative increase in magnitude is greater than for the SGVB model. Furthermore, we observe that a majority of the considered logs have missing measurement in the interval that extends over depths where the spike occurs, namely AI, DEN, NEU and CALI. These logs are thus not affiliated with the sudden increase in aleatoric uncertainty. Additionally, we see no change in the aleatoric uncertainty at depths where the measurements for these logs resume. However, we do observe that the AC log is vanishing exactly at depths where the spike extends. The remaining logs, i.e. GR and RMED, are seemingly constant over the interval and are thus not affiliated with the spike in aleatoric uncertainty.

In Figure 5.8 we consider an interval of depths with frequently occurring spikes in the aleatoric uncertainty, in particular for the MC Dropout model. We will not consider every fluctuation of the aleatoric variance within the interval but concentrate on the most notable changes.

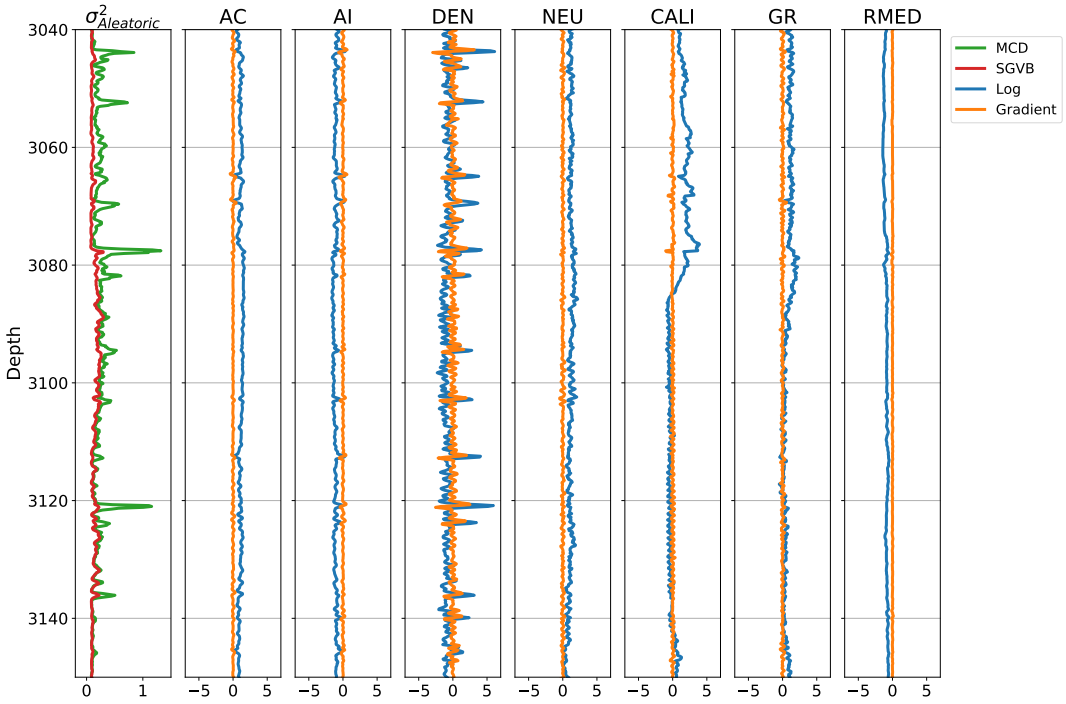


Figure 5.8: Aleatoric variance alongside a subset of the explanatory variables against depth for well 30/8-5 T2 over an interval with frequently occurring aleatoric spikes. In the leftmost plot, the aleatoric variance for the MC Dropout model is marked in green, while the SGVB model is marked in red. For the remaining figures, the logs are marked in blue, and the corresponding gradients in orange. The different logs are marked in the title of each plot.

We observe many depths within the considered interval where the aleatoric uncertainty suddenly changes. Moreover, we see that the aleatoric uncertainty provided by the MC Dropout model changes more notably than what is provided by the SGVB model.

At the start of the considered interval, we observe a spike in the aleatoric variance provided by the MC Dropout model, whereas the SGVB model provides no such response. We can also observe that at the same depth, there is a spike in the DEN log, as well as apparent noise in the AC, AI and GR logs. There are also slight changes in the NEU and CALI logs, whereas the RMED log is seemingly constant.

A few meters deeper, roughly around 3050 m, we observe another spike in aleatoric uncertainty provided by the MC Dorpout model. Similar to the shallower spike, the SGVB model does not provide a response. At the same depths, we observe a spike in the DEN log and sudden, although small, changes in the AC, AI, NEU, CALI, and GR logs. Moreover, the RMED log is seemingly constant.

At depths slightly shallower than 3080 m we observe another sudden increase in the aleatoric uncertainty provided by the MC Dropout model. For the SGVB model, the provided uncertainty increases slightly, albeit not to the extent provided by the MC Dropout model. At this depth, we observe a sudden increase in the DEN log and a relatively big decrease in the CALI log. Moreover, we observe that the AC

and GR logs experience a steady increase over the interval that extends over the spike in aleatoric uncertainty. However, this can not be seen as noise.

Slightly deeper than 3120 m we observe yet another spike in the aleatoric uncertainty. At the same depths, we also observe a sudden increase in the DEN log and a slight, sudden increase in AI. The remaining logs provide no interesting features.

We observe that the DEN log is quite noisy over the entire interval we consider and that there are depths in which the noise is more distinct than elsewhere. The above-mentioned spikes in aleatoric variance are all situated at depths where the noise in DEN is particularly high. For the remaining spikes in DEN, we observe an accompanying response in the aleatoric variance. Moreover, the response is greater for the variance provided by the MC Dropout than by the SGVB model. These fluctuations are, however, not to the extent of the above-described spikes, but we still observe that the aleatoric variance respond to noise in the DEN log.

In Figure 5.9 we consider an interval of depths where the aleatoric uncertainty is notably elevated, namely at the deeper end of the well. Interestingly, the elevated uncertainty remains high over a longer interval, unlike the above-described spikes.

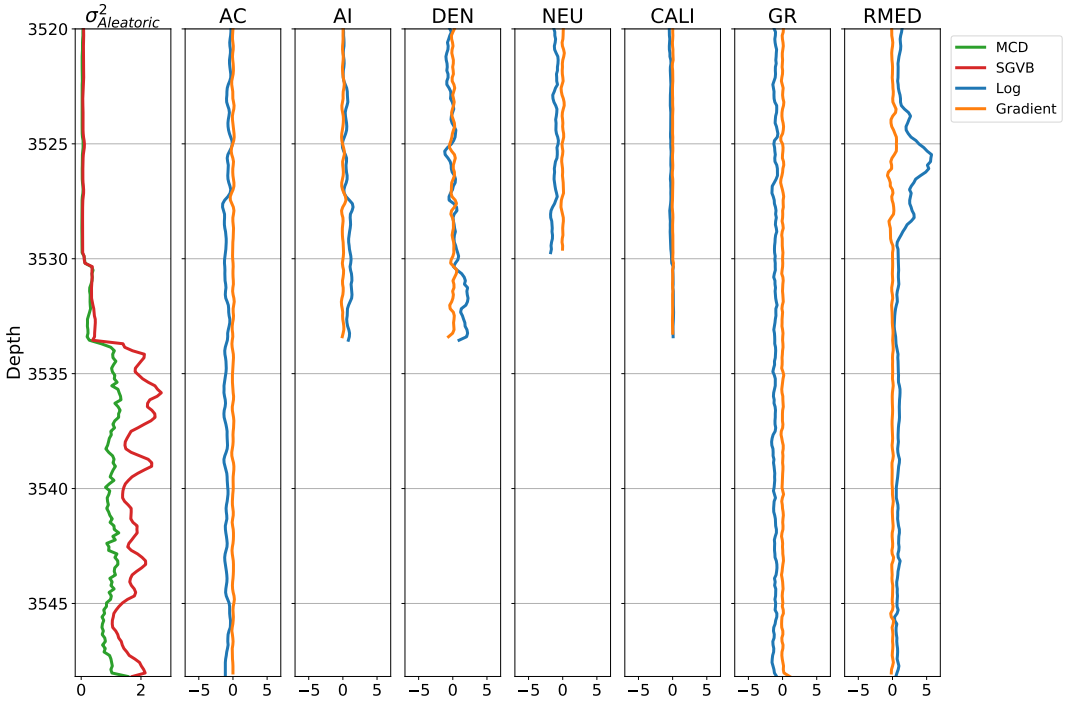


Figure 5.9: Aleatoric variance alongside a subset of the explanatory variables against depth for well 30/8-5 T2 in the interval of elevated aleatoric uncertainty at the end of the well. In the leftmost plot, the aleatoric variance for the MC Dropout model is marked in green, while the SGVB model is marked in red. For the remaining figures, the logs are marked in blue, and the corresponding gradients in orange. The different logs are marked in the title of each plot.

We observe that the aleatoric uncertainty provided by the models behave similarly. However, we see that the SGVB model is more sensitive than the MC Dropout model. We observe a slight increase in aleatoric uncertainty at ~ 3520 m, followed by a more notable increase. Unlike the spike above, there are no significant features of the AC log, so the change in uncertainty is attributed to the other logs. The same holds for the RMED log. We observe that the first increase

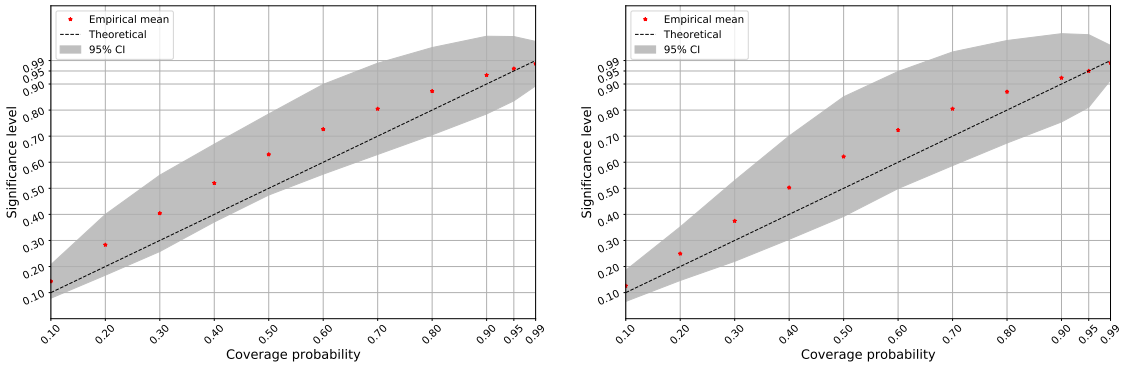
occurs at the depth where the measurements of the NEU log disappears. Moreover, we see that the subsequent increase in uncertainty occurs where the AI, DEN and CALI logs start missing. We observe that the missing measurement remains missing until the end of the interval, which might explain why the elevated uncertainty levels remain high for the remaining part of the interval. We also note an increase in the RMED log in the range $3525 - 3530\,\text{m}$, although not an accompanying change in the aleatoric uncertainty. However, the observed increase in RMED can not be seen as noise.

We have seen three intervals where the aleatoric uncertainty possesses particular attributes and analyzed the explanatory variables to investigate whether the estimated aleatoric uncertainty accounts for uncertainty in the data. From the above findings, we have seen that the aleatoric uncertainty provided by both models respond to noise in the explanatory variables, in particular, the MC Dropout model. Furthermore, the aleatoric uncertainty increases when the logs start missing and remains high if the logs remain missing. However, the aleatoric uncertainty remains seemingly constant at depths where the measurements resume, except when the AC log resume after the spike in aleatoric uncertainty around 3000 m. Moreover, it seems as if the increase in aleatoric uncertainty depends on the number of missing logs, where a greater number of missing logs result in a greater increase in uncertainty.

5.4 Quantitatively Evaluating the Predictive Uncertainty

A set of calibration curves are constructed for each model we consider to quantitatively evaluate the quality of the uncertainty estimates provided by the models. Due to the structure of the dataset, we construct a single calibration curve for each well in the test set. To aggregate the results, we present a single calibration curve using the empirical mean coverage probability across the test wells and provide a 95% credible interval around the mean. This allows one to reason about the uncertainty of the estimated uncertainty, at least to the extent of the wells in the test set. Separate calibration curves for all the test wells are found in Appendix D.

The aggregated calibration curves for the MC Dropout models are shown in Figure 5.10, where the homoscedastic and heteroscedastic models are shown in the left and right panel, respectively. The mean coverage probability is marked with a red star for every significance level, and the corresponding credible interval is seen as the gray, shaded area around the mean. A perfectly calibrated model coincides with the black, dotted line.



(a) Homoscedastic model

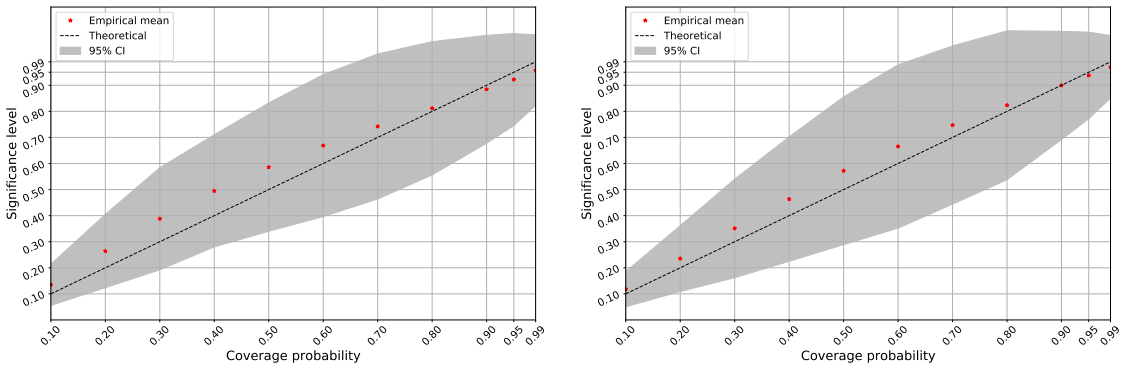
(b) Heteroscedastic model

Figure 5.10: Calibration plot across wells for the (a) homoscedastic and (b) heteroscedastic MC Dropout models. The mean empirical coverage (3.13) across the wells are shown as red stars, and a 95% credible interval is shown in grey. The diagonal, dotted line represents a perfectly calibrated model.

We observe that both models are well-calibrated, although not perfectly. The mean calibration across wells are above the diagonal for almost all significance levels for both models, showing that the uncertainty is over-estimated. There is no clear distinction between the homoscedastic and heteroscedastic model, other than the variability in calibration across wells is slightly greater for the heteroscedastic model, shown by the broader credible interval across the mean coverage probability. Moreover, we observe that the diagonal line corresponding to a perfectly calibrated model falls inside the credible interval around the mean calibration for both models.

The aggregated calibration curves for the SGVB models are shown in Figure 5.11, where the homoscedastic and heteroscedastic model is shown in the left and right panel of the figure, respectively. The mean coverage probability across the wells is marked as a red star for every significance level, and a 95% credible interval around

the mean is shown in gray. The black dotted line represents a perfectly calibrated model.



(a) Homoscedastic model

(b) Heteroscedastic model

Figure 5.11: Calibration plot across wells for the (a) homoscedastic and (b) heteroscedastic SGVB models. The mean empirical coverage (3.13) across the wells are shown as red stars, and a 95% credible interval is shown in grey. The black dotted line represents a perfectly calibrated model.

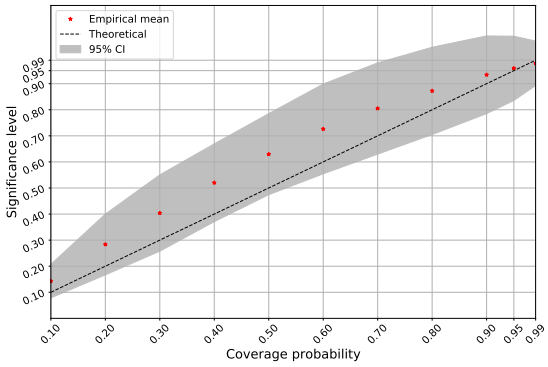
We observe that both models are well-calibrated, with a mean calibration that fits close to the diagonal line representing a perfectly calibrated model. Moreover, we observe that the diagonal line falls inside the credible interval for both the homoscedastic and heteroscedastic model.
Compared to the MC Dropout models in Figure 5.10, the mean calibration of the SGVB models is closer to the diagonal line. Consequently, the uncertainty estimates provided by the SGVB models are more properly estimated than the uncertainty estimates provided by the MC Dropout models. However, we observe that the across well variability is greater for the SGVB than the MC Dropout models, represented by the broader credible intervals. Hence, the mean calibration across wells is more uncertain for the SGVB models.

In the next part of this section, we will investigate the effect of modelling and including the aleatoric component to the total predictive uncertainty.

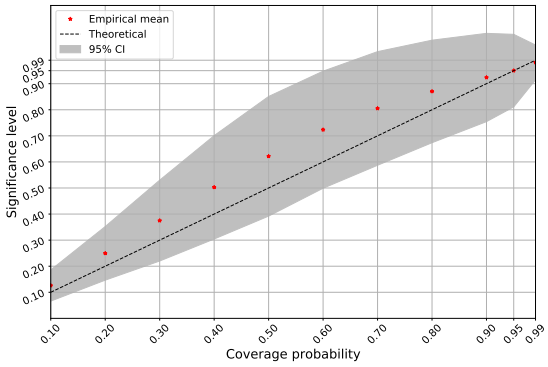
5.4.1 Effect of Modelling the Aleatoric Uncertainty

A set of calibration curves based on the epistemic uncertainty alone are constructed to look into the effect of modelling the aleatoric uncertainty. The resulting calibration curves are then aggregated, just as for the previous part of this section. By looking at the differences in calibration based on the epistemic and total predictive uncertainty, we can investigate the effect of modelling the aleatoric uncertainty.

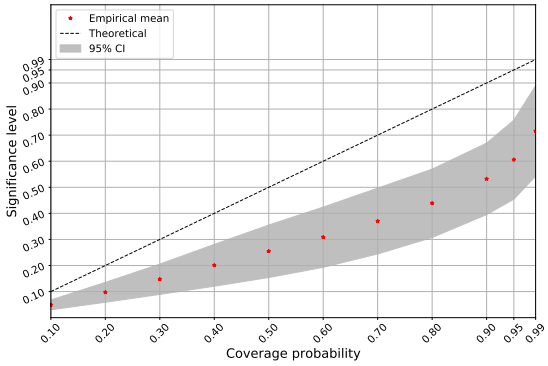
Figure 5.12 below shows the aggregated calibration curves for the MC Dropout models based on the total predictive uncertainty (Figures 5.12a and 5.12b) and the epistemic uncertainty alone (Figures 5.12c and 5.12d). The homoscedastic model is shown in the left column (Figures 5.12a and 5.12c), while the heteroscedastic model is shown in the right column (Figures 5.12b and 5.12d) of the figure.



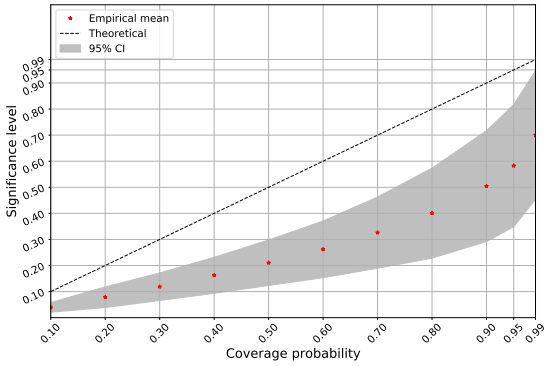
(a) Total predictive uncertainty



(b) Total predictive uncertainty



(c) Epistemic uncertainty alone

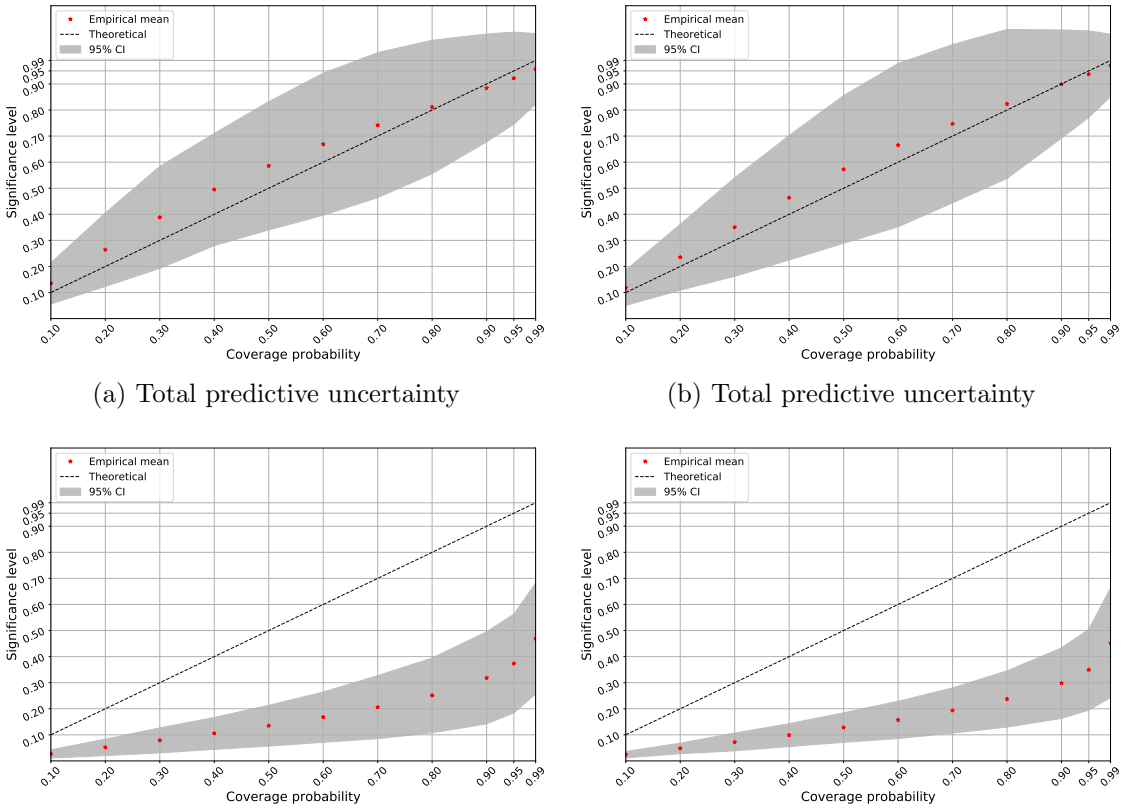


(d) Epistemic uncertainty alone

Figure 5.12: Calibration plots for a homoscedastic (left) and heteroscedastic (right) MC Dropout model based on (a)-(b) the total predictive uncertainty and (c)-(d) the epistemic uncertainty alone. The mean empirical coverage (3.13) across the wells are shown as red stars, and a 95% credible interval is shown in grey. A perfectly calibrated model is represented by the diagonal, dotted line.

We observe that the calibration curves based on the epistemic uncertainty alone are much different from the calibration curves based on the total predictive uncertainty for both models, particularly in the intermediate range of significance levels. The curves are under the diagonal line, showing that the uncertainty is under-estimated. Moreover, we observe that the differences between total and epistemic calibration are more notable for the heteroscedastic model.

In Figure 5.13 below we see the aggregated calibration curves for the SGVB models based on the total predictive uncertainty (Figures 5.13a and 5.13b) and the epistemic uncertainty alone (Figures 5.13c and 5.13d). The left column (Figures 5.13a and 5.13c) corresponds to the homoscedastic model, while the right column (Figures 5.13b and 5.13d) corresponds to the heteroscedastic model.



(a) Total predictive uncertainty

(b) Total predictive uncertainty

(c) Epistemic uncertainty alone

(d) Epistemic uncertainty alone

Figure 5.13: Calibration plots for a homoscedastic (left) and heteroscedastic (right) SGVB model based on (a)-(b) the total predictive uncertainty and (c)-(d) the epistemic uncertainty alone. The mean empirical coverage (3.13) across the wells are shown as red stars, and a 95% credible interval is shown in grey. A perfectly calibrated model is represented by the diagonal, dotted line.

As for the MC Dropout models, we clearly see that the calibration curves based on the epistemic uncertainty alone are very different from the calibration curves based on the total predictive uncertainty for both SGVB models. The curves are far below the diagonal line, showing that the uncertainty is highly under-estimated.

By including the aleatoric uncertainty, thus obtaining the total predictive uncertainty, the uncertainty estimates yields empirical coverage probabilities that are more similar to the significance levels of the corresponding confidence intervals. Consequently, the uncertainty estimates provided by the total predictive uncertainty are more suitable than the epistemic uncertainty for explaining the predictive uncertainty of the models. This highlights the importance of including the aleatoric uncertainty in quantifying the predictive uncertainty of the models for the dataset we consider. Moreover, the differences in the calibration curves imply that the aleatoric uncertainty dominates the total predictive uncertainty.

5.5 Epistemic Uncertainty and Training Set Size

To look into the convergence properties of the epistemic uncertainty with respect to the amount of observed data, we train all our models on different fractions of the entire training set. We consider evenly spaced fractions in the range $10 - 100\%$, and the different training sets are constructed by randomly sampling the entire training set. For each fraction of the training set, the epistemic uncertainty is aggregated by computing the mean standard deviation of the predictive distributions for all the samples in the entire test set. Using the same architecture, i.e. hypothesis space, for all the models allows to study the approximation uncertainty, i.e. the component of the epistemic uncertainty that represent the amount of observed data.

In Figure 5.14 we see the aggregated epistemic uncertainty for all the considered models against the different fractions of the training set. The homoscedastic MC Dropout model is shown in orange, while the heteroscedastic MC Dropout model is shown in blue. For the SGVB models, the homoscedastic model is marked in red, while the heteroscedastic model in marked in green.

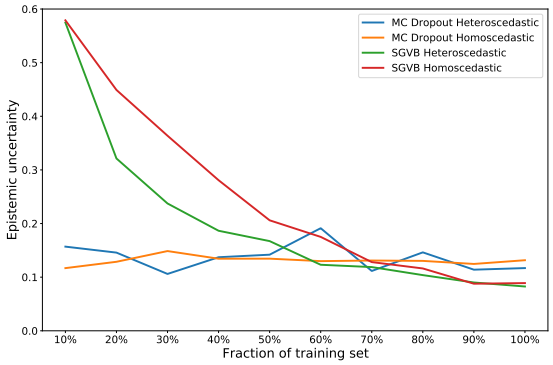


Figure 5.14: Epistemic uncertainty for all the considered models trained on varying sized training sets.

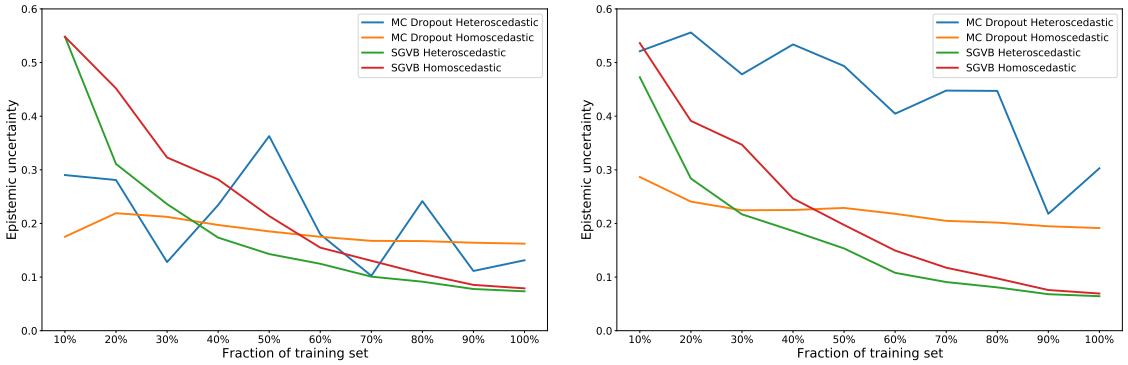
For the SGVB models, we see that the epistemic uncertainty decrease steadily and seemingly converge at 90% of the entire training set. This in is compliance with the stated property of the epistemic uncertainty being reducible with respect to new information. Moreover, we observe that the epistemic uncertainty for the heteroscedastic model is generally lower than for the homoscedastic model, particularly before the epistemic uncertainty has converged.

Regarding the MC Dropout models, the situation is quite different. We observe that the aggregated epistemic uncertainty is seemingly constant across fractions of the dataset and thus independent of the size. The epistemic uncertainty is, in this case, not reducible with respect to new information, which is not compliant with the stated property of the epistemic uncertainty. Note that the presented results in Figure 5.14 consider the architecture described in Section 4.4 with a 10% dropout rate.

Referring back to section 3.1, the posterior predictive distribution of the parameters in the MC Dropout models, and thus the epistemic uncertainty, is directly related to the dropout rate used to train the models.

In Figure 5.15 we see the epistemic uncertainty against varying fractions of the training set using a dropout rate of 30% (Figure 5.15a) and 50% (Figure 5.15b).

We observe that the epistemic uncertainty of the MC Dropout models highly depends on the dropout rate, while the SGVB models are seemingly unaffected.



(a) 30% dropout rate

(b) 50% dropout rate

Figure 5.15: Epistemic uncertainty provided by all the considered models for varying sized training sets with (a) 30% and (b) 50% dropout rates.

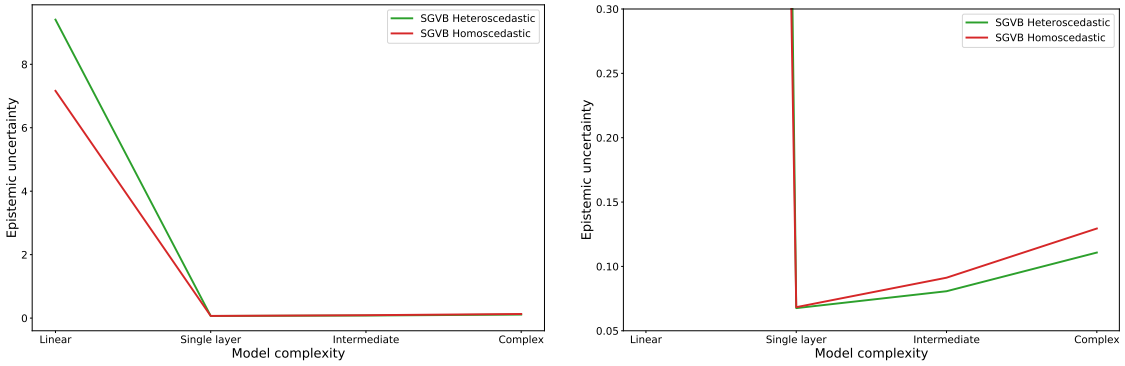
The epistemic uncertainty of the MC Dropout models seem to increase with the dropout rate, and it is apparent that the difference between the homoscedastic and heteroscedastic model increase as well. Furthermore, by increasing the dropout rate, we observe that the epistemic uncertainty starts to decrease to a greater extent when observing greater fractions of the training set, making the epistemic uncertainty reducible with respect to new information. It is also apparent that the epistemic uncertainty of the heteroscedastic models is oscillating to a greater extent than for the homoscedastic model.

For the SGVB models, we observe the same steadily decreasing trend in the epistemic uncertainty as in Figure 5.14, where the heteroscedastic model has a lower level of epistemic uncertainty than the homoscedastic model for all the fractions of the entire training set. Moreover, we observe that the epistemic uncertainty of the models converges to approximately the same level for all the considered dropout rates, unlike the MC Dropout models.

5.6 Epistemic Uncertainty and Model Complexity

To investigate the epistemic uncertainty with respect to the hypothesis space, we consider a set of different architectures and study how the epistemic uncertainty changes. Letting the models observe the same amount of data during training allows studying how the model uncertainty is affected by changing the hypothesis space because the resulting approximation uncertainty is expected to be approximately constant[2].

In Figure 5.16 the aggregated epistemic uncertainty over the full test set is plotted against an increasingly big hypothesis space for the homoscedastic (red) and heteroscedastic (green) SGVB models. In the left panel of the figure, we see the full range of the resulting epistemic uncertainty, and in the right panel, the values on the vertical axis are restricted to be able to observe the differences between the models. The different hypothesis spaces, or equivalently, model complexities, are described in section 4.4.2.



(a) Entire range of values on the vertical axis (b) Restricted range of values on the vertical axis

Figure 5.16: Epistemic uncertainty for varying model complexities for the SGVB models. In (a) the entire range of values are plotted, whereas a restricted range of values are plotted in (b). The homoscedastic model is marked in red, while the heteroscedastic model is marked in green.

From Figure 5.16a we observe that the epistemic uncertainty provided by the linear models are far greater than the other complexities, where the heteroscedastic model result in a greater value than the homoscedastic model. By increasing the hypothesis space from a linear to a single layer model, the epistemic uncertainty is explained away and is seemingly constant over the remaining model complexities. Consequently, we can say that a single layer model is sufficient for explaining away the model uncertainty, whereas a linear model does not suffice.

In fig. 5.16b, where the values on the vertical axis are restricted to a smaller range, we observe that the minimum value is obtained for the single-layer model and that the values increase slightly with increasing model complexity. Moreover, we see that the heteroscedastic model obtain lower values of the epistemic uncertainty for these model complexities, unlike the linear hypothesis space where the homoscedastic model achieves a lower value. However, both the increase and differences between

[2]We do, however, expect the approximation uncertainty to increase when using more complex models, as they require more training data to converge

the models are marginally small compared to the transition from a linear to a single-layer model.

Chapter 6

Discussion

A discussion of the results is presented in this chapter. The discussion follows the same structure as the results, with separate sections for each part. The chapter is concluded by discussing real-world consequences of the findings and proposals for future work.

Referring back to Section 1.5, the goal of this thesis was first and foremost to construct and apply Bayesian neural networks that can quantify their predictive uncertainty in a problem setting provided by the company this thesis is written in collaboration with. Furthermore, the predictive uncertainty of the BNNs was decomposed into its aleatoric and epistemic components, and the aleatoric uncertainty was modelled in a homoscedastic and heteroscedastic way. Having decomposed the predictive uncertainty into the different components, the aleatoric uncertainty was qualitatively validated by investigating whether it responds to noise in the data. The uncertainty estimates were quantitatively evaluated using calibration curves to make sure the provided uncertainty is reliable. Regarding the epistemic uncertainty, an analysis was carried out to investigate the different sources of such, namely the approximation and model uncertainty.

Two methods for constructing BNNs were implemented, namely Monte Carlo Dropout and Stochastic Gradient Variational Bayes. Both methods rely on variational inference for updating the distributions of the parameters of the neural networks. Furthermore, the MC Dropout method was expanded from its original formulation in [16, 17], where the aleatoric uncertainty of the data is specified as a hyper-parameter. The novelty in the applied MC Dropout method lies in the fact that the negative log-likelihood function was used as a loss function to model the aleatoric uncertainty using data. This has previously been done using the SGVB method [20] and follows directly from the probabilistic interpretation of the model specification.

6.1 Methods for Obtaining Bayesian Neural Networks

This part of the discussion considers the presented results in Section 5.1 and 5.2.

The results from the presented prediction curves for both methods in Figures 5.1 and 5.4 show that the predictions follow the ground truth values well, although there are some discrepancies. The differences between the methods are practically negligible, except for a greater discrepancy between the predicted and true values in the outer extremities of the well for the SGVB method. Regarding the uncertainty provided by the credible intervals around the predictions, the methods seem to behave similarly, in particular for the spike we observe at $\sim 3000\,\text{m}$ and the interval of elevated aleatoric uncertainty in the deeper end of the well for the heteroscedastic models. However, we can observe that the epistemic uncertainty provided by the SGVB models is lower than what is provided by the MC Dropout models. The epistemic uncertainty directly follows from the posterior predictive distribution of the parameters of the models. For the MC Dropout models, this depends solely on the dropout rate used to train the models. On the contrary, the posterior predictive distribution for the parameters of the SGVB models is learned by updating the prior to obtain the posterior during training. One can thus argue that the epistemic uncertainty provided by the SGVB model is more flexible than the MC Dropout models, as the posterior distribution of the parameters of the latter is set as a hyper-parameter rather than being learned from data.

There is a clear difference in the credible intervals around the predictions provided by the homoscedastic and heteroscedastic models. The difference arises from the fact that the homoscedastic models provide a constant aleatoric uncertainty, while the heteroscedastic models allow the aleatoric uncertainty to vary across samples. Hence, there will be intervals where the aleatoric uncertainty provided by the homoscedastic model is over-estimated and other intervals where the uncertainty is under-estimated. We observe that the empirical coverage probability is greater than expected from the 95% credible interval for the MC Dropout models, and the corresponding credible intervals are too wide. For the heteroscedastic model, the empirical coverage probability is only slightly higher than expected. This might imply that the constant aleatoric uncertainty is over-estimated for a majority of the predictions provided by the homoscedastic model. For the SGVB models, the empirical coverage probabilities provided by both models are comparable, although lower than expected from the probability level of the credible intervals.

The ground-truth values for the uncertainty of the predictions are not available, and we have to rely on calibration curves to investigate whether the estimated uncertainty is appropriately estimated. However, this will only apply to the entire test set and does not allow to investigate whether the uncertainty is properly estimated at particular depths of interest. We can, however, perform a qualitative analysis of the aleatoric component of the predictive uncertainty to confirm or reject potentially occurring intervals of elevated aleatoric uncertainty.

Regarding convergence of the training, we see from Figure 5.3 that both MC Dropout models converge properly. Moreover, we observe that the magnitude of the homoscedastic loss is notably higher than the heteroscedastic loss after the first epoch. The difference in magnitude might be a result of poor initialization of the parameter representing the aleatoric uncertainty in the homoscedastic model, resulting in a high value of the loss in the first pass through the training set.

Referring back to Section 4.4.2 the parameter is initialized by drawing samples from a standard normal distribution. In contrast, the parameters of the model predicting the heteroscedastic aleatoric uncertainty are initialized with the robust He-initialization. For the SGVB models, we see in Figure 5.6 that neither the homoscedastic nor heteroscedastic model has converged properly after ten epochs. Referring back to Section 3.2 we have that each parameter in the SGVB models contains two learnable parameters, namely the variational parameters representing the mean and standard deviation of the corresponding distributions. Consequently, the SGVB models contain twice the number of learnable parameters than the MC Dropout models, and the optimization requires more iterations to converge. There are, however, no clear distinction between the homoscedastic and heteroscedastic models, although the heteroscedastic model contains even more parameters than the homoscedastic model[1].

In terms of predictive performance across all the wells in the test set, we see that the heteroscedastic models consistently outperform the homoscedastic models in both performance metrics we consider. There is, however, one exception for well 25/4-10 S, where the homoscedastic MC Dropout model outperforms the corresponding heteroscedastic model in terms of the MAE metric. This indicates that there are greater discrepancies between the true and predicted values for the heteroscedastic than the homoscedastic MC Dropout model for that particular well. Looking at the prediction curve for well 25/4-10 S in Figure C.1, we do observe a greater discrepancy for the heteroscedastic model, in particular in the shallower end of the well. Moreover, we observe that the majority of the samples in the well are missing, causing the corresponding performance metrics to be exceptionally high for all the considered models.

The predictive performance over the entire test set shows that the heteroscedastic models perform better than the homoscedastic models for both methods. This is expected by the consistently better well-wise performance. The better performance of the heteroscedastic models highlights the importance of allowing the aleatoric uncertainty to vary across samples in the dataset we consider. Not only will the heteroscedastic models provide a richer representation of the uncertainty, they also improve the predictive performance over the homoscedastic models.

We observed that the values of the performance metrics for the SGVB models are generally higher than the corresponding values for the MC Dropout models. This implies that the MC Dropout models provide a better fit to the dataset than the SGVB models and explains the greater discrepancy between the true and predicted value of the SGVB models in the outer extremities of the presented well, as mentioned above. Moreover, the performance metrics provided by the SGVB models are associated with greater uncertainty. The predictive performance is thus better for the MC Dropout models, although the performance metrics provided by the SGVB models are more uncertain. This can be seen in conjunction with the fact that both MC Dropout models have converged after ten epochs, while the SGVB models are still converging (see figs. 5.3 and 5.6).

Regarding the model-wise comparison, we see that the homoscedastic MC Dropout models outperform SGVB and the deterministic architecture in terms of both performance metrics. For the heteroscedastic models, we have that the SGVB model outperform the other models in terms of the MSE metric, while the deterministic model performs better in terms of the MAE metric. However, we do observe that both probabilistic models outperform the deterministic in terms of

[1] due to the auxiliary model predicting the aleatoric log-variance

the MSE metric. This implies that there are greater discrepancies between the predicted and true values for the deterministic than the probabilistic models, as the MSE metric is more sensitive to discrepancies than MAE. This is because the former calculates the squared error, while the latter calculates the error in absolute terms (See (3.10)).

The ground-truth values of the predictive uncertainty are not available, and we can thus not measure how well the uncertainty is estimated. Referring back to Section 2.1.2, the aleatoric uncertainty aims to capture the uncertainty in the data. By comparing the estimated aleatoric uncertainty with the explanatory variables in the dataset, we can investigate whether the aleatoric uncertainty emerges from the data. We performed such an analysis in Section 5.3, and the results are discussed in the next section.

6.2 Qualitative Analysis

From the qualitative analysis of the aleatoric uncertainty presented in Section 5.3, we see that the aleatoric uncertainty provided by the heteroscedastic models respond to noise and missing values in the explanatory variables, albeit to different extents. This is precisely what we expect from the definition of aleatoric uncertainty, which represents uncertainty in the data. The MC Dropout model is seemingly more sensitive than the SGVB model for sudden changes in the explanatory variables, while the aleatoric uncertainty provided by the SGVB models is more affected when logs are missing. Regarding missing logs, we see no particular response in the aleatoric uncertainty when measurements resume, except from the anomaly around 3000 m (see Figure 5.7). Ideally, we would have elevated uncertainty levels where logs are missing and a similar decrease when measurements resume. However, we only observe elevated uncertainty levels in the transition where measurements are vanishing and only a single case of the opposite effect in the transition from missing to resuming measurements.

We investigated three intervals along the depths of the presented well where the aleatoric uncertainty changes (see Figures 5.7 to 5.9). Around 3000 m depth, we observe a spike in the aleatoric uncertainty provided by both models. A majority of the logs are missing before and after the spike and are thus not affiliated with the response. However, the AC log vanishes exactly at the depths in which the spike occur. Thus, the aleatoric uncertainty is very sensitive to changes in the AC log, and the MC Dropout model is more affected than SGVB. Note that the only logs present around the depths at which the spike occur are GR and RMED. The great magnitude of the spike might be a combination of all the missing logs, and that there are only two logs present for explaining the response. This should be expected, as the predictions have more degrees of freedom in the case of more missing explanatory variables. Moreover, from a physical perspective, the GR and RMED logs are not nearly enough to accurately infer the value of the response variable ACS. In this case, we cannot say that the missing logs that extend across the anomaly do not contribute to the increased uncertainty. However, we see that these logs are missing before and after the spike and that the aleatoric uncertainty return to its seemingly constant pre-spike value when the measurements of the AC log resume. From a physical perspective, the most important variable to the regression is AC because AC and ACS are typically strongly correlated except in the presence of pore fluids. Consequently, the above-described sensitivity of the aleatoric uncertainty to the AC log is expected.

In the interval around $3040 - 3150$ m, we see that the aleatoric uncertainty is quite noisy. This can be explained by the noise in the explanatory variables, in particular in the DEN log. We observe that the aleatoric uncertainty spikes at depths where the noise in DEN is abnormally high. Moreover, we see that the aleatoric uncertainty is even more affected when the other logs experience sudden changes on top of the noisy DEN log. In general, spikes in the DEN log are evidence of so-called stringers, short intervals of particularly hard rocks. During drilling, an encounter with a stringer cause problems and difficulties when taking measurements, and the resulting data is typically noisy. The rapid spikes we observe in the DEN log in Figure 5.8 might be a result of rapidly occurring stringers, causing the measurements to be noisy and uncertain.

In the deeper end of the well, we observe an interval of elevated aleatoric uncertainty, and the high levels are associated with missing logs. Moreover, we

observe that the aleatoric uncertainty increase in a step-wise manner. At first, we see a slight increase in uncertainty at depths where the NEU log vanishes. Subsequently, there is an even greater increase in the uncertainty levels at depths where the logs for AI, DEN and CALI disappear simultaneously. Consequently, we can say that the amount of which the aleatoric uncertainty increase depends on the number of missing logs. This is expected, as the predictions should be more uncertain when more explanatory variables are missing. The missing logs remain missing until the end of the well, and the aleatoric uncertainty responds by remaining high. We observe that the SGVB model is more affected than the MC Dropout model in the case of missing logs.

By performing a qualitative analysis as presented in Section 5.3, we can validate whether the attributes of the aleatoric uncertainty originates from noise in the data at particular depths. However, this type of analysis will not provide a numerical value for how accurately the uncertainty is estimated. The calibration curves presented in Section 5.4 allows to state how well the uncertainty is estimated numerically, and a discussion of the results follows in the next section.

6.3 Quantitatively Evaluating the Predictive Uncertainty

This section is a discussion of the results regarding the evaluation of the uncertainty estimates presented in Section 5.4.

From the aggregated calibration curves in Figures 5.10 and 5.11 we see that all the models provide uncertainty estimates that are close to a perfectly calibrated model represented by the diagonal line. Moreover, we see that both methods yield credible intervals with empirical coverage probabilities that are slightly above the significance levels, meaning that the credible intervals are broader than what is expected by the corresponding confidence intervals. Consequently, the uncertainty estimates are over-estimated, causing the models to yield under-confident predictions. In a practical scenario where the predictions are passed to a human for further inspection, if the provided uncertainty is above some specified threshold, these models would slow down the automation process as potentially correct predictions would be passed to a human more rapidly than if the uncertainty estimates were perfectly calibrated. This is, however, more conservative than the opposite situation, where potentially erroneous predictions would pass through without human inspection more rapidly, causing potential harm.

Although the uncertainty is slightly over-estimated, the perfectly calibrated model falls within the 95% credible interval around the mean empirical coverage probability for all the models we consider. Hence, there is a 95% probability that the uncertainty estimates provided by the models are perfectly calibrated.

For the MC Dropout model, we observe that the cross-well variability in empirical coverage is larger for the heteroscedastic than the homoscedastic model. This means that the uncertainty of the uncertainty estimates are greater for the heteroscedastic model. Regarding the SGVB models, there is no apparent difference in the credible interval around the empirical coverage. However, we observe that the credible intervals are notably wider for the SGVB models than the MC Dropout models. This implies that the estimated uncertainty provided by the SGVB models are more uncertain than the estimates provided by the MC Dropout models, as the cross-well variability is greater for the former.

Compared to the MC Dropout models, we see that the mean empirical coverage lies closer to the diagonal line for the SGVB models. Hence, the SGVB method provides more trustworthy uncertainty estimates than MC Dropout. This might be explained by the fact that the variance of the posterior distribution of the parameters in the SGVB models are learned during training, while the variance is set as a hyper-parameter in the MC Dropout models. Consequently, the variance of the parameters in the SGVB models are more flexible than for the MC Dropout models.

Furthermore, we observed that the calibration curves based on the epistemic uncertainty alone are located far below the diagonal line. Consequently, the uncertainty is highly under-estimated, and the predictions are thus over-confident. This highlights the importance of modelling the aleatoric uncertainty as part of the predictive uncertainty, as uncertainty measures based on the epistemic uncertainty alone yields over-confident predictions. In Section 1.2 we described a non-Bayesian method for obtaining the predictive uncertainty in neural networks by training an ensemble of such, each having a different initialization of the parameters. The method proposed in [38] estimates the predictive uncertainty by the sample variance of the predictions provided by the ensemble. This approach is analogous to estimating the epistemic uncertainty in the Bayesian neural networks

we consider. Consequently, the predictive uncertainty provided by the deep ensemble on the dataset we consider is likely to under-estimate the uncertainty of the predictions. Moreover, the authors in [38] does not provide any evaluation of the uncertainty estimates in a regression setting[2], although they state that the deep ensemble yields high-quality uncertainty estimates.

The differences in the calibration curves based on the total predictive uncertainty and the epistemic uncertainty alone can be explained intuitively by using the calibration curves based on the total predictive uncertainty in Figures 5.10 and 5.11 in conjunction with the prediction curves presented in Figures 5.1, 5.2, 5.4 and 5.5 and the remaining prediction curves in Appendix C. From the prediction curves, we observe that the aleatoric uncertainty constitutes the majority of the total predictive uncertainty, as the credible interval based on the total predictive uncertainty is far wider than the credible intervals based on the epistemic uncertainty alone. Moreover, we observe from the calibration curves based on the total predictive uncertainty that the uncertainty is properly estimated. By excluding the aleatoric uncertainty, we are removing a majority of the uncertainty, and the resulting credible intervals become much tighter. As a result, the uncertainty estimates based on the epistemic uncertainty alone becomes highly under-estimated, and the resulting calibration curves will lie below the diagonal.

The fact that there is a greater uncertainty attached to estimating the uncertainty provided by the SGVB models needs to be seen in conjunction with how the distribution of the parameters is obtained and the distribution of the response variable across the dataset. As mentioned above, the variance of the parameters provided by the MC Dropout method is essentially set as a hyper-parameter while being learned during training for the SGVB method. One can thus argue that the variance provided by the SGVB method is more reliable than provided by MC Dropout, even though the provided uncertainty estimates are properly calibrated for both methods. The response variable is distributed differently across the wells in the dataset (see Section 4.2), which might result in inductive bias in the predictions. Using this in conjunction with the greater cross-well variability in calibration for the SGVB models might imply that the SGVB models are better at capturing out-of-distribution test data represented by the different distributions across the wells and that the MC Dropout models under-estimate the cross-well variability.

In the next section, we will discuss the result of the analysis of the epistemic uncertainty with respect to the size of the dataset and the complexity of the models.

[2]they do however provide such an evaluation in a classification setting

6.4 Analysis of the Epistemic Uncertainty

In this section, a discussion of the analysis of the epistemic uncertainty is presented. The results are found in Section 5.5 and 5.6. In the first subsection, we will discuss the convergence properties of the epistemic uncertainty with respect to dataset size, i.e. the approximation uncertainty, before we discuss the effect of model complexity in the following subsection. The analysis performed to investigate whether the estimated epistemic uncertainty is reducible in terms of obtaining more knowledge, as stated in [12, 27, 30].

6.4.1 Epistemic Uncertainty and Training Set Size

In Section 2.1.2 we define the epistemic uncertainty in terms of approximation and model uncertainty. By definition, we have that the approximation uncertainty depends on the amount of observed data, while the model uncertainty depends on the size of the hypothesis space, or equivalently, the complexity of the model. To separate the approximation from the model uncertainty, we train models with identical complexities for a varying fraction of the entire training set and estimate the epistemic uncertainty over the entire test set.

From Figure 5.14 we observe that the epistemic uncertainty provided by the SGVB models clearly decrease when observing a greater fraction of the dataset during training. In contrast, the MC Dropout models provide a seemingly constant epistemic uncertainty across all fractions. Moreover, we observe that the heteroscedastic SGVB model decrease in a greater fashion than the corresponding homoscedastic model until the epistemic uncertainty converge to the approximately same value for both models after observing 90% of the entire training set. Furthermore, we observe in Figure 5.15 that the estimated value of the epistemic uncertainty provided by the MC Dropout models highly depends on the dropout rate, which is excepted from the definition of the distribution of the parameters in the network 3.6. Moreover, we observe that as the dropout rate increase, the amount of which the epistemic uncertainty decreases when observing more of the training data increase. This can be explained using the fact that a greater dropout rate results in a greater variance of the posterior distribution of the parameters of the MC Dropout models, such that there is more epistemic uncertainty to explain away by observing more data. As expected from the definition of the SGVB models in Section 3.2, these models are seemingly unaffected by the dropout rate.

The notable difference in behaviour between the models can be seen in conjunction with how the posterior distribution of the parameters of models is obtained. As mentioned above, the variance of the parameters in the MC Dropout models is essentially set as a hyper-parameter, which is a major drawback of the MC Dropout method. On the contrary, the variance of the parameters is learned during training for the SGVB method. However, we do observe from Figure 5.14 that MC Dropout yields values of the epistemic uncertainty that is comparable to the converged values provided by the SGVB models, although the former are slightly higher. Moreover, we observe that the total predictive uncertainty is properly calibrated for the MC Dropout models in Figure 5.10. However, from Figure 5.15 we observe that the MC Dropout models provide greater values for the epistemic uncertainty when using greater dropout rates. Consequently, the resulting estimates of the total predictive uncertainty are likely to be even more

over-estimated than what is provided by the MC Dropout models using a 10% dropout rate.

In the next part of this section, we will discuss the results regarding the analysis of the epistemic uncertainty concerning model complexity.

6.4.2 Epistemic Uncertainty and Model Complexity

From Figure 5.16a we observe that the epistemic uncertainty is notably higher in the case of a linear model compared to the other complexities we consider. Moreover, we see that the epistemic uncertainty is explained away when using a model with a single hidden layer. The values remain seemingly constant when increasing the model complexity further. Consequently, we can say that a single layer model is sufficient for explaining away the model uncertainty.

For the linear model, the hypothesis space is essentially restricted to the space provided by the explanatory variables. From our findings, we see that this hypothesis space is too small for containing the optimal model because the model uncertainty is not explained away until we extend the hypothesis space to include non-linear models. This is expected by the preliminary analysis presented in Section 4.3, where we observe that a linear model is not sufficient for explaining the variability in the dataset. By increasing the model complexity from a linear to a single layer neural network, the capacity of the model is dramatically increased, and likewise the size of the hypothesis space. It is apparent that the hypothesis space represented by the single-layer model contains the optimal model, and there is thus no more model uncertainty to resolve by further increasing the size of the hypothesis space.

Although the epistemic uncertainty is seemingly constant when increasing the model complexity from a single layer to more complex models, there are some differences which can be seen in Figure 5.16b where the values on the vertical axis are restricted to a smaller range. We see that both models achieve a minimum value for the single-layer model and that the epistemic uncertainty marginally increase when increasing the model complexity. Fitting a more complex model requires more data, and we can thus expect that the approximation uncertainty will increase when increasing model complexity. Even though the model uncertainty might be constant, we observe that the epistemic uncertainty increases when increasing the model complexity further. We do, however, observe that the increase is only marginal compared to the decrease when going from a linear to a single layer model, and we can say that the model uncertainty is practically constant for the larger models. Consequently, we say that the marginal increase in the epistemic uncertainty is likely attributed to the approximation uncertainty.

We see that the epistemic uncertainty for the SGVB models behaves as expected. By obtaining more information in terms of data and the size of the hypothesis space, the epistemic uncertainty is explained away. However, it is hard to separate the model from the approximation uncertainty as more complex models require more data. It is impossible to obtain a linear model using the MC Dropout method. We can thus not observe how the epistemic uncertainty provided by the MC Dropout models respond to changes in model complexity when going from a linear to a single-layer model. Regarding dataset size, the epistemic uncertainty provided by the MC Dropout models is seemingly constant and highly dependent on the dropout rate.

It is important to note that the above-presented results and discussion are not agnostic in terms of neither the considered models nor the dataset. The presented results are only a single sample of such, and we have to conduct the same type of analysis to a broad range of problems to state that the results are general and not specific to the models and dataset used in our analysis.

The above-presented results account for the epistemic uncertainty in the model and the aleatoric uncertainty in the data. There are, however, other sources of uncertainty in deep learning that can be analyzed. The initialization of the parameters of the neural networks is typically performed by drawing samples from a probability distribution. Moreover, the optimization method used to train the models is typically stochastic, particularly in the case of mini-batch optimization and the above described Adam algorithm used to train our models. Consequently, there will be uncertainty attached to the process of initializing the parameters and training the models. This uncertainty is however not accounted for in the presented results.

In real-world situations where deep learning is applied in a decision-making process, rather than relying on potentially over-confident predictions provided by a traditional neural network with no measure of uncertainty, our Bayesian models will provide a measure of uncertainty to their predictions. Consequently, we can leverage the uncertainty provided by our models to make decision-making based on deep learning more trustworthy. The predictions can e.g. be passed to a human for manual inspection if the uncertainty is above some specified, context-dependent threshold or otherwise let the prediction pass through the automation process without human interference. We observe that the uncertainty estimates are appropriately estimated and can capture the epistemic and aleatoric components in a satisfying way. This allows investigating whether the uncertainty originates from the model or the data. However, there is a drawback in terms of how the uncertainty is obtained. The uncertainty estimates are obtained using a sampling approach (see Section 3.3), where we have to pass the same instance through the models multiple times to obtain a predictive distribution for that particular instance. For large models and datasets, this might become computationally expensive, albeit not infeasible. If the uncertainty measures are needed in real-time, e.g. if the models are applied in a context provided by an autonomous vehicle, the sampling approach may turn out to be too slow to work out in practice. If the models are applied in a context where the additional computations are not critical, e.g. as in the above-presented case study, the sampling approach is perfectly suitable.

This work has been very exciting and rewarding, but some things could have been done otherwise. In particular, the well-wise scaling of the response variable described in Section 4.2.1. Because the true value of the response variable is used in obtaining the corresponding z-score (2.9), it is not possible to return to the original scale of the variable without leveraging the true values. In practical situations where the response values are needed to make decisions, the preprocessing scheme applied in this thesis will not provide any value, as the predictions provided by the models are the z-scores of the response. However, it is important to note that the scheme does not affect the remaining analysis of the uncertainty. It is also important to note that the well-wise scaling avoids the flow of information between the training and test set because the different datasets are split well-wise. Consequently, we avoid data leakage that would yield results that are not representative of the actual performance of the models on unseen data. On the contrary, if we perform an inverse

scaling of the z-scores of the predictions, we would need to leverage the true values of the response variable of the instances we are predicting. The true values of the response are only available in a test situation and will result in data leakage as the true values are required to return to the original scale. An approach for dealing with this drawback is only to scale the explanatory variables, leaving the response variable in its original scale.

Moreover, the initial plan of the thesis was to carry out a similar procedure for quantifying the predictive uncertainty in a classification setting. During our work on the regression problem, we stumbled upon interesting things we wanted to investigate, e.g. the analysis of the epistemic uncertainty concerning the amount of data and model complexity. We decided that the thesis would become more interesting if we chose to do a more thorough analysis on a single problem, rather than simply quantifying the predictive uncertainty in two widely different problems. The choice was made relatively late in the semester, and it would have spared much time thinking about how to perform a similar analysis in the classification setting if the choice of abandoning the problem was made earlier.

There are many interesting extensions to the work carried out in this thesis, and a list with suggestions for future work follows with a brief description and motivation.

i. **Uncertainty and model performance**
   This can be investigated by considering a range of threshold values for the uncertainty and discard instances that yield a predictive uncertainty above this threshold from the test set from which the performance metrics (3.10) are calculated. From the loss functions (see eqs. (3.4), (3.5) and (3.7)) it is expected that larger errors are accompanied with greater uncertainty. However, we did not perform such an analysis, and it would interesting to look into the connection between the model performance and the predictive uncertainty.

ii. **Additional sources of uncertainty in deep learning**
   As mentioned in the above discussion, there is likely to be uncertainty attached to the initialization and optimization of neural networks. However, we did not include this into the total predictive uncertainty. To look into the uncertainty due to the optimization method, we propose to train multiple networks with identical configurations multiple times and compute the variance of the corresponding predictions. The uncertainty due to stochastic initialization might be investigated by training multiple networks with different initializations and then similarly computing the predictions' variance. It is necessary to train the networks using a deterministic optimization algorithm to separate the uncertainty due to the initialization.

iii. **Deep ensembles**
   In the above discussion, we criticized the deep ensemble approach for obtaining predictive uncertainty by showing that the epistemic uncertainty alone yields highly under-estimated uncertainty on our dataset. We propose to extend the method in [38] to include the aleatoric uncertainty in the modelling process, thus obtaining a richer representation of the uncertainty.

iv. **Recalibration**
The authors in [37] propose a method shown to provide calibrated uncertainty estimates by training an auxiliary recalibration model that creates a dataset for training the final prediction model. Our models provide properly calibrated uncertainty estimates, albeit slightly over-estimated, resulting in under-confident predictions. It would be interesting to extend the applied methods with an auxiliary recalibration model as proposed in [37], and see if the resulting models provide perfectly calibrated uncertainty estimates.

v. **Single-pass predictive distribution**
A drawback with obtaining the predictive distribution in this thesis is the sampling approach, which may be too slow in a practical context where uncertainty estimates are required in real-time. In such a context, it would be very valuable to have a model that can provide uncertainty estimates by doing a single forward pass of the network for each instance, thus superseding the sampling. In [60] the authors propose a method for doing so, although the method builds upon deterministic neural networks rather than Bayesian neural networks. It might be possible to extend the Bayesian neural networks upon the ideas in [60] to obtain a predictive distribution by a single forward pass.

vi. **Multi-task learning**
To improve upon the generalization abilities of the models, in particular, due to the inductive bias posed by the different distributions of the dataset across wells, we propose to extend the models to allow for multitask learning [9]. Multitask learning aims to predict multiple tasks, where each task, in this case, can be seen as the response variable in each well. We believe that this approach will improve the generalization abilities of the models and potentially yield better estimates of the uncertainty. A method applied to a dataset with a similar structure is proposed in [47].

vii. **Transfer learning**
The idea behind transfer learning is to improve generalization abilities by relieving the induction bias posed by dataset shift, thus relaxing the i.i.d assumption (2.1). In network-based transfer learning, a model is pre-trained on a general dataset before the parameters are fine-tuned to a more specific dataset [55]. In this case, one can transfer knowledge from the general to the specific dataset and better capture the differences. We propose to extend our models to allow for transfer learning. In our case, we would use the entire training set to pre-train the models and then fine-tune the parameters by using a subset of the instances in the test set for each well. Similar to the above proposed multitask extension, this will result in a single model for each well. This will, however, not allow making predictions over the full extent of the well, as some of the instances are being used to fine-tune and thus train the models.

Chapter 7

Closing Remarks

Two methods for constructing Bayesian neural networks was applied and extended to quantify and decompose the predictive uncertainty in a regression problem provided by the company this thesis was written in collaboration with. The dataset consists of borehole data gathered from 34 wells from offshore Norway, and the problem aims to predict the acoustic log for S-waves based on geophysical measurements. The predictive uncertainty was decomposed into the epistemic uncertainty attached to the models, and the aleatoric uncertainty in the data. Furthermore, the aleatoric uncertainty was modelled in a homoscedastic and heteroscedastic way, where the uncertainty is constant and allowed to vary across samples in the dataset, respectively. The ground-truth values of the uncertainty were clearly not available, and we relied on calibration curves to numerically evaluate how the uncertainty was estimated. The calibration curves only applies to the full test set, and does not allow us to investigate if the uncertainty at particular depths are properly estimated. By using the definition of the aleatoric uncertainty, we did however perform a qualitative analysis which showed that the heteroscedastic aleatoric uncertainty provided by the two methods responds to noise and measurement errors in the explanatory variables at particular depths, albeit to different extents. Furthermore, an analysis of the epistemic uncertainty was carried out to investigate whether it is reducible in terms of obtaining more information about the system being modelled. The analysis was carried out in terms of training set size and model complexity, the latter in conjunction with a preliminary analysis showing that a linear model is not sufficient for explaining the variability in the data.

The predictions provided by all the models are shown to fit the true values well, and the heteroscedastic models consistently outperform the corresponding homoscedastic models for both methods. This highlights the importance of allowing the uncertainty in the data to vary across samples. Moreover, the performance metrics for the MC Dropout models are better than the corresponding metrics for the SGVB models, and the latter are more uncertain. Consequently, the MC Dropout provides a better fit to the data than SGVB.

The provided uncertainty was shown to be properly estimated for all the considered models, although they are slightly over-estimated. Consequently, the predictions are under-confident, and will slow down the automation process if the models are used in a practical situation. We also observed that the uncertainty estimates based on the epistemic uncertainty alone are highly under-estimated, highlighting the importance of modelling the aleatoric uncertainty.

From the analysis of the epistemic uncertainty, we see that the SGVB models behaves as expected and the epistemic uncertainty is shown to be reducible by obtaining more information. The values of the epistemic uncertainty provided by the homoscedastic and heteroscedastic models decrease steadily until converging to approximately the same value when increasing the training set. On the contrary, the MC Dropout models yield a seemingly constant epistemic uncertainty when observing more data. This can be explained by the way in which the variance of the parameters of the models are obtained, and henceforth the epistemic uncertainty. For the MC Dropout method, the variance of the parameters are essentially set as an hyper-paramer, and require proper tuning of the dropout rate to provide properly estimated uncertainties. The dropout rate that yields proper uncertainty estimates might not coincide with the dropout rate that yields optimal performance in terms of a performance metric when doing a grid-search to tune the hyper-parameters, and one have to be careful when using MC Dropout to estimate the predictive uncertainty. This is major drawback of the MC Dropout method. The variance of the parameters of the SGVB method are learned during training, and the epistemic uncertainty provided by the SGVB models are thus more flexible than MC Dropout.

For the SGVB models we observe that the hypothesis space represented by a linear model is not sufficient for explaining the variability in the data, resulting in a high value for the epistemic uncertainty. This is compliant with the preliminary analysis showing that a linear model is not sufficient for explaining the data. The epistemic uncertainty is explained away by increasing the complexity to a single layer neural network, and remains seemingly constant by further increasing the complexity of the models. Consequently, we say that a single layer model is sufficient for explaining away the model uncertainty, whereas a linear model do not suffice.

From our findings we conclude that the SGVB method is superior to the MC Dropout method in terms of quantifying predictive uncertainty, although the latter yields lower values for the considered performance metrics than the former. We state this because the epistemic uncertainty of the SGVB method is more flexible than MC Dropout, which needs proper tuning of the dropout rate to provide accurate estimates of the uncertainty. This is also reflected in the expected behaviour of the epistemic uncertainty provided by the SGVB models.

Appendix A

Well-wise distribution of target variable

A.1 Training set

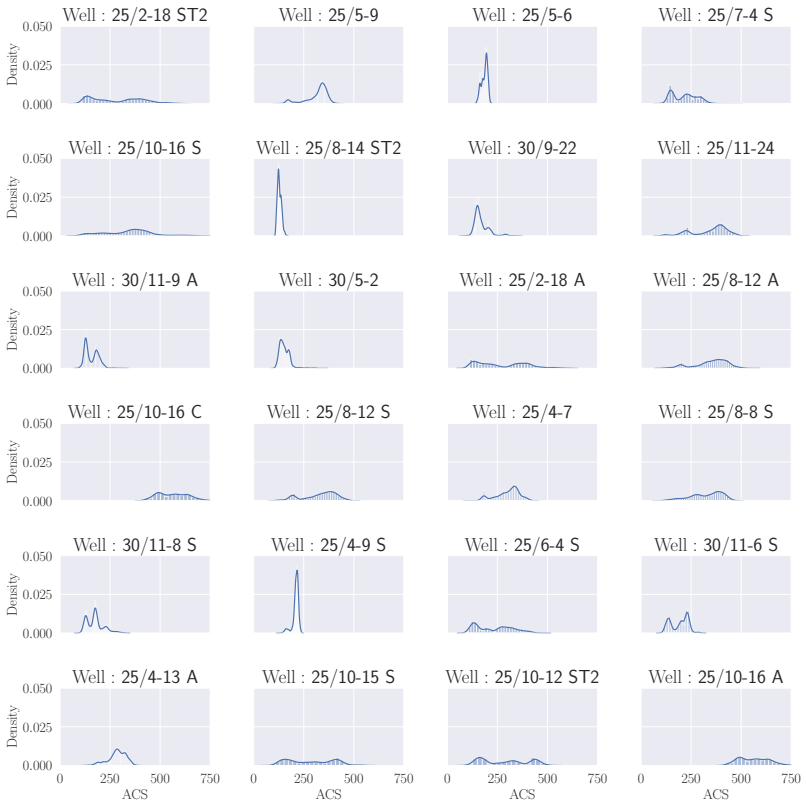


Figure A.1: Well-wise distribution of target variable in the training set. The name of the well is marked in the title of each figure.

A.2 Validation set

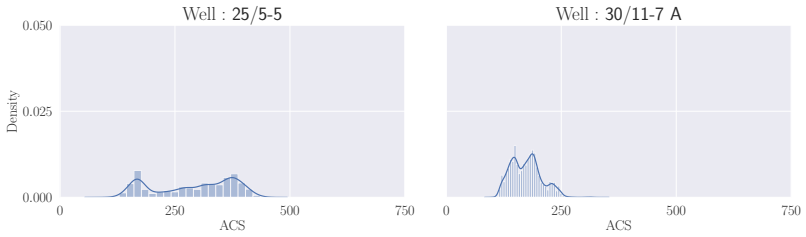


Figure A.2: Well-wise distribution of target variable in the validation set. The name of the well is marked in the title of each figure.

A.3 Test set

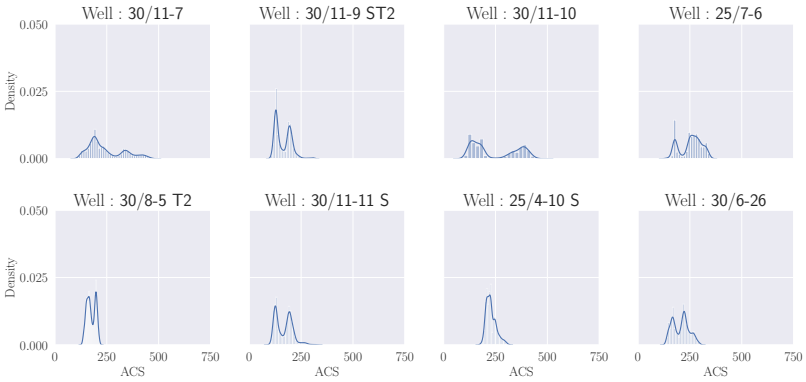


Figure A.3: Well-wise distribution of target variable in the test set. The name of the well is marked in the title of each figure.

Appendix B

Analytical derivation of the ELBO loss

The ELBO loss is defined as

$$\mathcal{L}(q_\lambda(\boldsymbol{\theta})) = \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] - D_{KL}[q_\lambda(\boldsymbol{\theta})||p(\boldsymbol{\theta})].$$

We will derive the two components of the ELBO loss separately, and start off with the KL-divergence between the variational distribution and the prior for the latent variables.

$$\begin{aligned} D_{KL}[q_\lambda(\boldsymbol{\theta})||p(\boldsymbol{\theta})] &= \int_{\mathbb{R}^K} q_\lambda(\boldsymbol{\theta}) \log \frac{q_\lambda(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} d\boldsymbol{\theta} \\ &= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\big[\log \frac{q_\lambda(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\big] \\ &= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}[\log q_\lambda(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})] \end{aligned} \tag{B.1}$$

By the mean-field assumption on the variational distribution and the prior distribution of the latent variable, we have

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{K} p(\theta_i) = \prod_{i=1}^{K} \frac{1}{\sqrt{2\pi\bar{\sigma_i}^2}} e^{-\frac{1}{2}\left(\frac{\theta_i - \bar{\mu_i}}{\bar{\sigma_i}^2}\right)^2}$$

$$q_\lambda(\boldsymbol{\theta}) = \prod_{i=1}^{K} q_{\lambda_i}(\theta_i) = \prod_{i=1}^{K} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2}\left(\frac{\theta_i - \mu_i}{\sigma_i}\right)^2},$$

where $\lambda_i = (\mu_i, \rho_i)$ and $\sigma_i = \log(1 + e^{\rho_i})$.

The log-likelihood of the distributions are thus

$$\begin{aligned} \log p(\boldsymbol{\theta}) &= \sum_{i=1}^{K} \log p(\theta_i) = \sum_{i=1}^{K} -\frac{1}{2}\log 2\pi\bar{\sigma_i}^2 - \frac{1}{2}\left(\frac{\theta_i - \bar{\mu_i}}{\bar{\sigma_i}}\right)^2, \\ \log q_\lambda(\boldsymbol{\theta}) &= \sum_{i=1}^{K} \log q_{\lambda_i}(\theta_i) = \sum_{i=1}^{K} -\frac{1}{2}\log 2\pi\sigma_i^2 - \frac{1}{2}\left(\frac{\theta_i - \mu_i}{\sigma_i}\right)^2. \end{aligned} \tag{B.2}$$

Inserting (B.2) into (B.1) we get the following expression for the KL-divergence

between the variational distribution and the prior for the latent variables.

$$\begin{aligned} D_{KL}[q_\lambda(\boldsymbol{\theta})||p(\boldsymbol{\theta})] &= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\sum_{i=1}^{K} \log q_{\lambda_i}(\theta_i) - \log p(\theta_i)\right] \\ &= \frac{1}{2}\mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\sum_{i=1}^{K} -\log 2\pi\sigma_i^2 - \left(\frac{\theta_i-\mu_i}{\sigma_i}\right)^2 + \log 2\pi\bar{\sigma}_i{}^2 + \left(\frac{\theta_i-\bar{\mu}_i}{\bar{\sigma}_i}\right)^2\right] \\ &= \frac{1}{2}\mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\sum_{i=1}^{K} 2\log\frac{\bar{\sigma}_i}{\sigma_i} - \frac{1}{\sigma_i^2}(\theta_i-\mu_i)^2 + \frac{1}{\bar{\sigma}_i{}^2}(\theta_i-\bar{\mu}_i)^2\right] \\ &= \frac{1}{2}\sum_{i=1}^{K}\left[2\log\frac{\bar{\sigma}_i}{\sigma_i} - \frac{1}{\sigma_i^2}\mathbb{E}_{q_{\lambda_i}(\theta_i)}[(\theta_i-\mu_i)^2] + \frac{1}{\bar{\sigma}_i{}^2}\mathbb{E}_{q_{\lambda_i}(\theta_i)}[(\theta_i-\bar{\mu}_i)^2]\right]. \end{aligned} \tag{B.3}$$

The term $\mathbb{E}_{q_{\lambda_i}(\theta_i)}[(\theta_i-\mu_i)^2]$ is simply σ_i^2, as the distribution of the latent variable θ_i is distributed with mean μ_i and standard deviation σ_i. This is by definition of the variance of a random variable.

To further simplify the expression for the KL-divergence, we need to rewrite the last term in (B.3).

$$\begin{aligned} \mathbb{E}_{q_i}[(\theta_i-\bar{\mu}_i)^2] &= \mathbb{E}_{q_i}[\theta_i^2 - 2\theta_i\bar{\mu}_i + \bar{\mu}_i{}^2] \\ &= \mathbb{E}_{q_i}[\theta_i^2] - 2\mu_i\bar{\mu}_i + \bar{\mu}_i{}^2 \\ &= \mathrm{Var}_{q_i}[\theta_i] + \mathbb{E}_{q_i}[\theta_i]^2 - 2\mu_i\bar{\mu}_i + \bar{\mu}_i{}^2 \\ &= \sigma_i^2 + \mu_i^2 - 2\mu_i\bar{\mu}_i + \bar{\mu}_i{}^2 \\ &= \sigma_i^2 + (\mu_i-\bar{\mu}_i)^2, \end{aligned} \tag{B.4}$$

where $\mathrm{Var}_{q_i}[\theta_i]$ is the variance of the latent variable θ_i over the variational distribution $q_i = q_{\lambda_i}(\theta_i)$, such that $\mathrm{Var}_{q_i}[\theta_i] = \mathbb{E}_{q_i}[\theta_i^2] - \mathbb{E}_{q_i}[\theta_i]^2$ by the variance decomposition formula.

By inserting (B.4) into (B.3) we get the final expression for the KL-divergence.

$$D_{KL}[q_\lambda(\boldsymbol{\theta})||p(\boldsymbol{\theta})] = \frac{1}{2}\sum_{i=1}^{K}\left[2\log\frac{\bar{\sigma}_i}{\sigma_i} + \left(\frac{\sigma_i}{\bar{\sigma}_i}\right)^2 + \left(\frac{\mu_i-\bar{\mu}_i}{\bar{\sigma}_i}\right)^2 - 1\right]$$

By assuming independent measurements, we can express the likelihood of the data as

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x_i},\boldsymbol{\theta}),$$

where N is the number of measurements. From the generative model in (4.2) we can express the log-likelihood of the data as follows.

$$\begin{aligned} \log p(\mathcal{D}|\boldsymbol{\theta}) &= \sum_{i=1}^{N}\log p(y_i|\boldsymbol{x_i},\boldsymbol{\theta}) \\ &= \sum_{i=1}^{N}\log\frac{1}{\sqrt{2\pi}g(\boldsymbol{x_i},\boldsymbol{\psi})}e^{-\frac{1}{2}\left(\frac{y_i-f(\boldsymbol{x_i},\boldsymbol{\phi})}{g(\boldsymbol{x_i},\boldsymbol{\psi})}\right)^2} \\ &= -\frac{N}{2}\log 2\pi - \sum_{i=1}^{N}\left[\log g(\boldsymbol{x_i},\boldsymbol{\psi}) - \frac{1}{2}\left(\frac{y_i-f(\boldsymbol{x_i},\boldsymbol{\phi})}{g(\boldsymbol{x_i},\boldsymbol{\psi})}\right)^2\right] \end{aligned} \tag{B.5}$$

The final expression for the expected log-likelihood is thus

$$\mathbb{E}_{q_\lambda(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] = \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\Bigg[-\frac{N}{2}\log 2\pi - \sum_{i=1}^{N}\Bigg[\log g(\boldsymbol{x_i},\boldsymbol{\psi}) - \frac{1}{2}\Bigg(\frac{y_i - f(\boldsymbol{x_i},\boldsymbol{\phi})}{g(\boldsymbol{x_i},\boldsymbol{\psi})}\Bigg)^2\Bigg]\Bigg]$$

The log-likelihood in (B.5) is the general form that holds for input-dependent, heteroscedastic noise, governed by $g(\boldsymbol{x_i},\boldsymbol{\psi})$. For a homoscedastic noise model, i.e where $g(\boldsymbol{x_i},\boldsymbol{\psi}) = \sigma\ \forall i$, the log-likelihood reduce to

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = -\frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(\boldsymbol{x_i},\boldsymbol{\phi}))^2$$

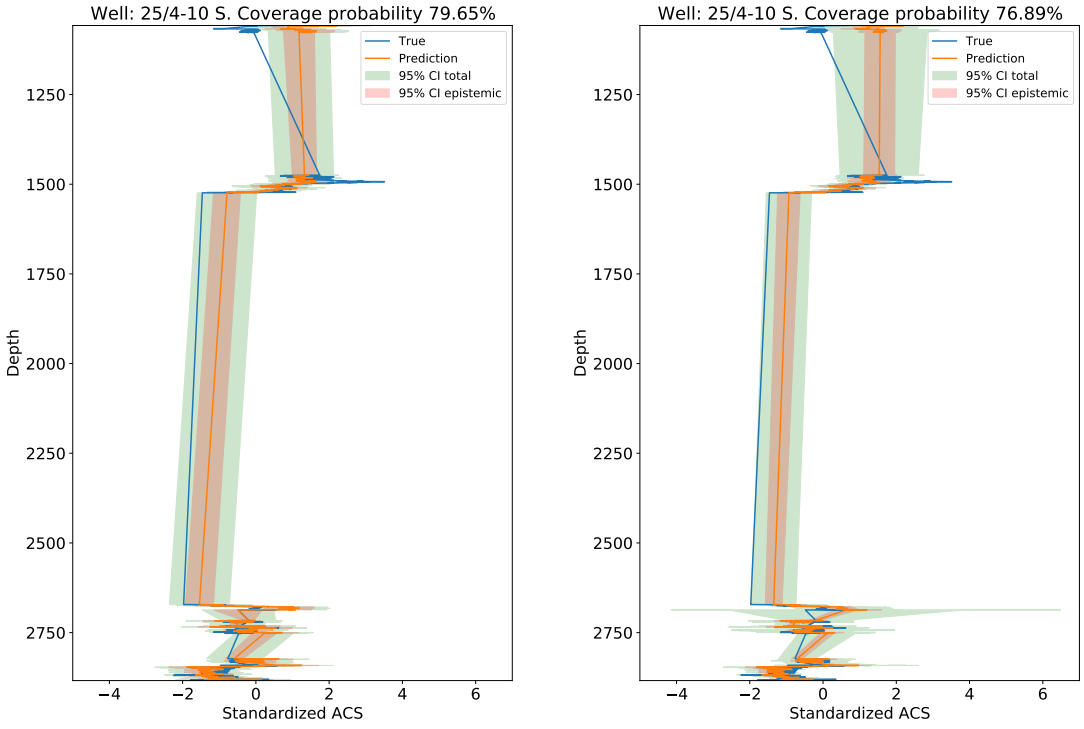
Appendix C

Wellwise predictions

This part of the appendix shows the predictions curves for all the wells in the test test. The first section considers the MC Dropout method, while the last section is concerned witht the SGVB method. The predictions are plotted against depth, and are equipped with 95% credibility intervals corresponding to the epistemic and total predictive uncertainty. The differences between the credibility intervals represents the aleatoric uncertainty.

Some of the wells contains missing values at specific depths to to poor measurements (see 4.2.1). The missing measurements can be seen as constant values in the below plots, where the ends of the interval of missing values are connected by a line-plot. Do not interpret these seemingly constant values as predictions by the model, they simply represent that the samples are missing.

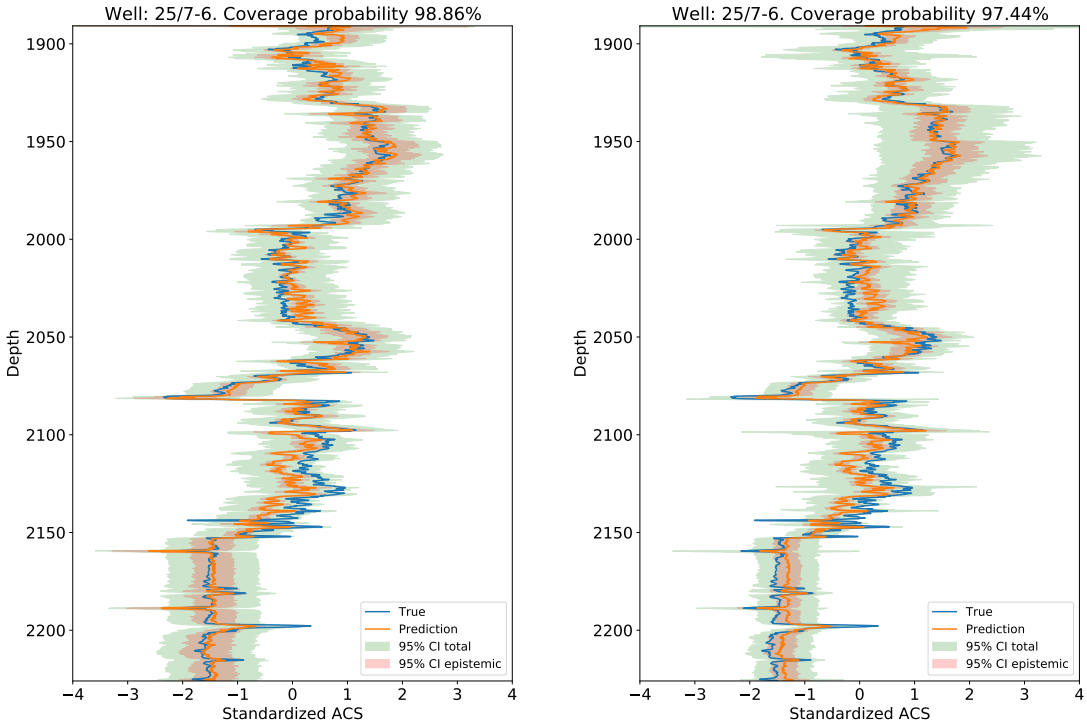
C.1 MC Dropout



(a) Homoscedastic model

(b) Heteroscedastic model

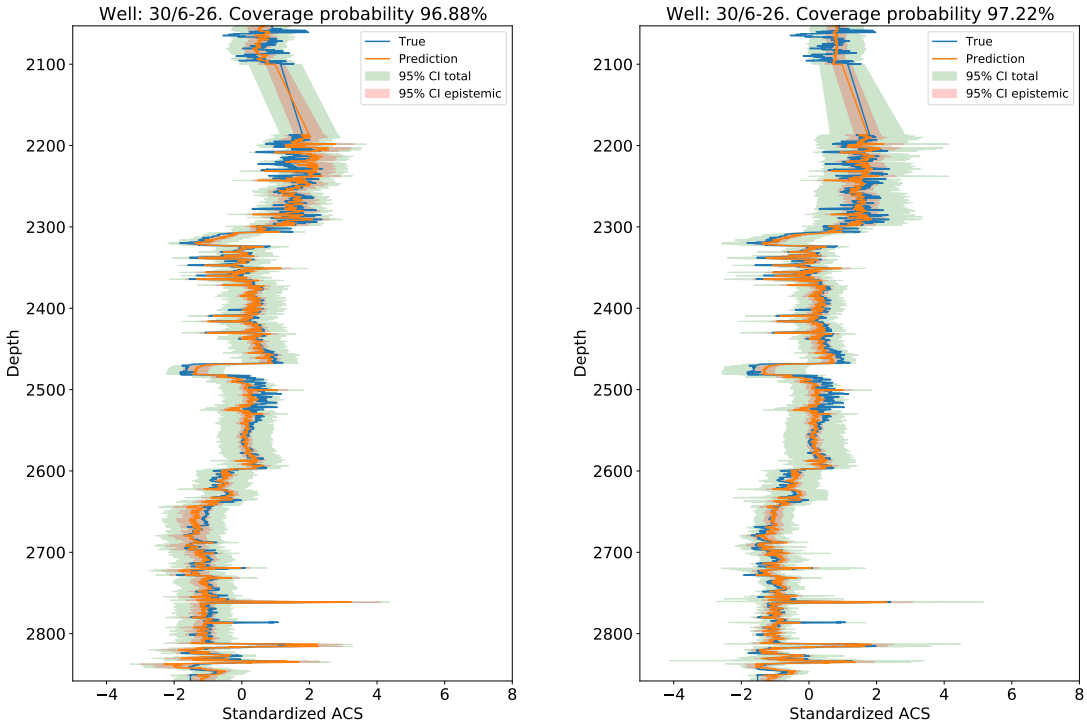
Figure C.1: Predictions and corresponding 95% credible intervals for well 25/4-10 S in the test set using a (a) homoscedastic and (b) heteroscedastic MC Dropout model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

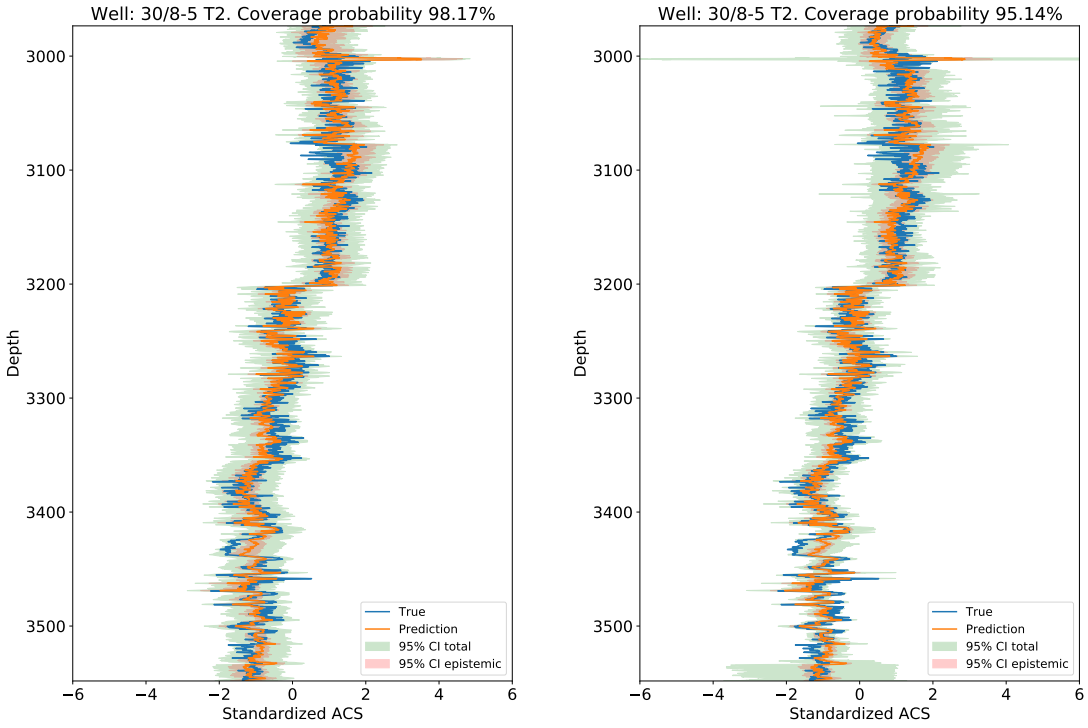
Figure C.2: Predictions and corresponding 95% credible intervals for well 25/7-6 in the test set using a (a) homoscedastic and (b) heteroscedastic MC Dropout model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

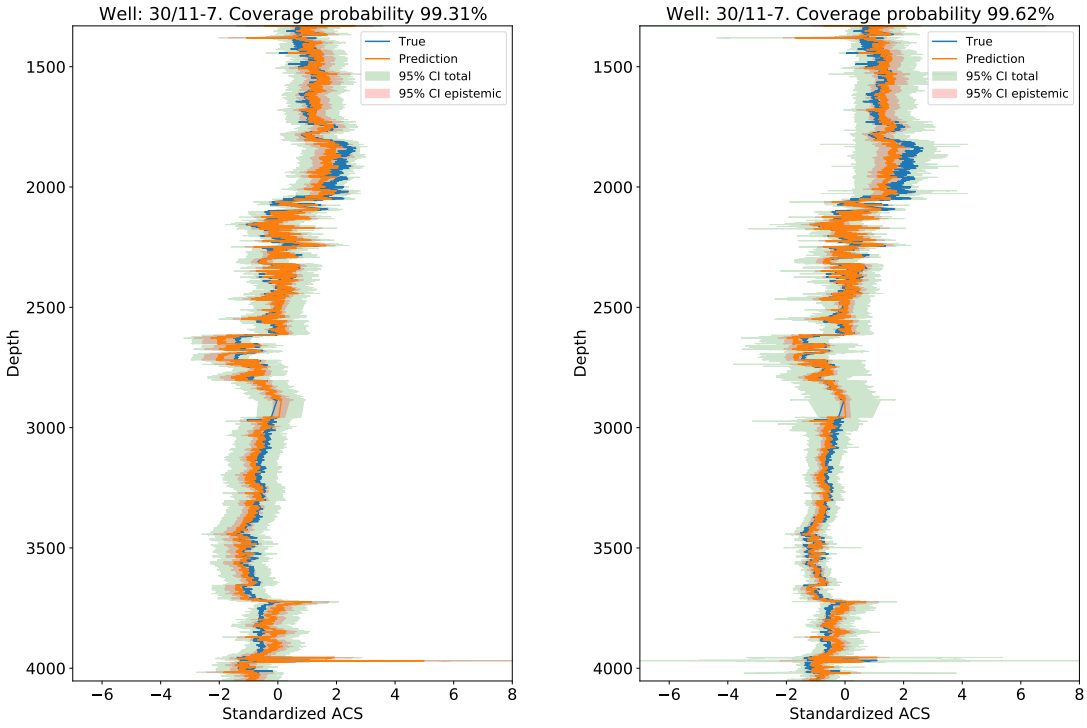
Figure C.3: Predictions and corresponding 95% credible intervals for well 30/6-26 in the test set using a (a) homoscedastic and (b) heteroscedastic MC Dropout model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

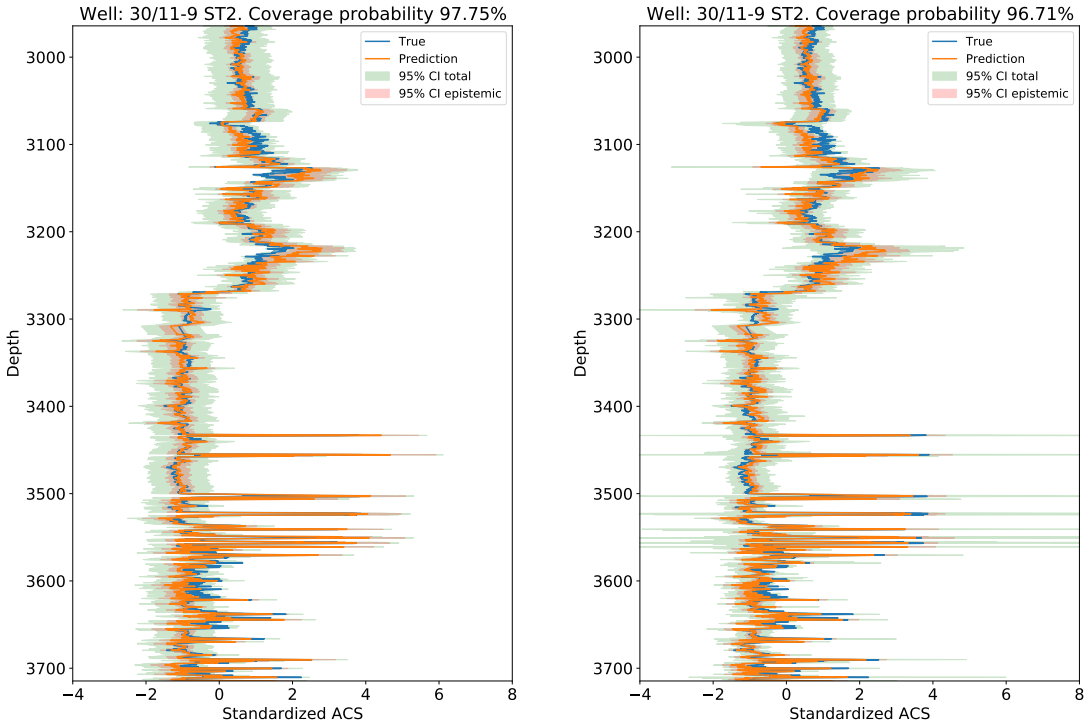
Figure C.4: Predictions and corresponding 95% credible intervals for well 30/8-5 T2 in the test set using a (a) homoscedastic and (b) heteroscedastic MC Dropout model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

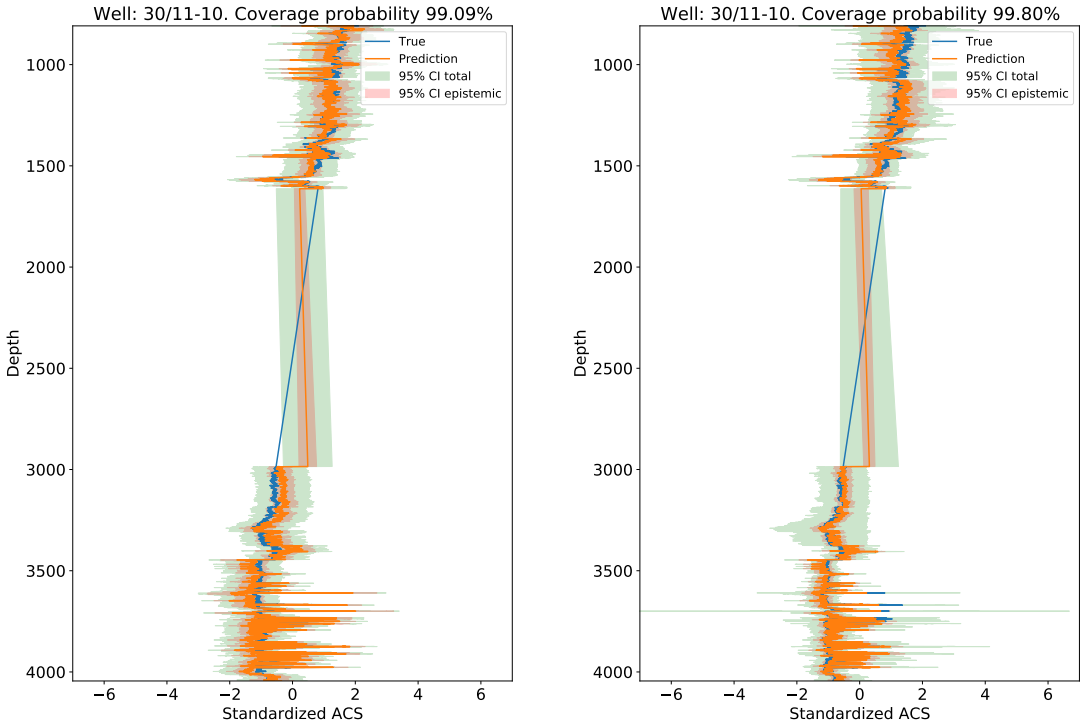
Figure C.5: Predictions and corresponding 95% credible intervals for well 30/11-7 in the test set using a (a) homoscedastic and (b) heteroscedastic MC Dropout model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

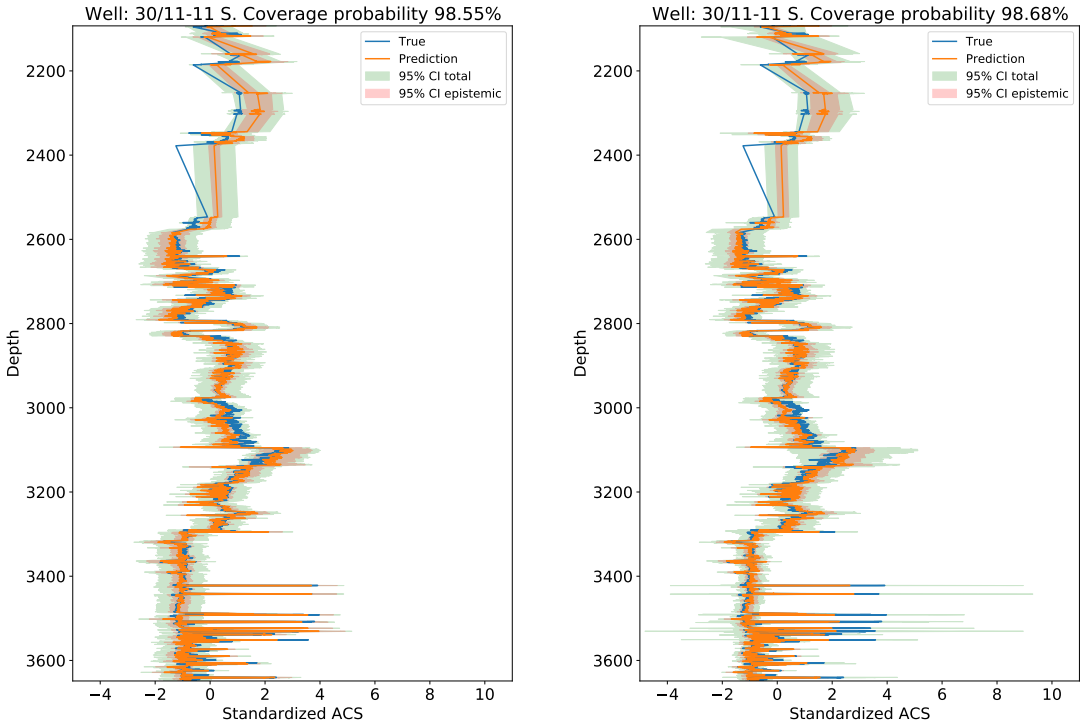
Figure C.6: Predictions and corresponding 95% credible intervals for well 30/11-9 ST2 in the test set using a (a) homoscedastic and (b) heteroscedastic MC Dropout model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

Figure C.7: Predictions and corresponding 95% credible intervals for well 30/11-10 in the test set using a (a) homoscedastic and (b) heteroscedastic MC Dropout model. The empirical coverage probability is marked in the title of each plot.

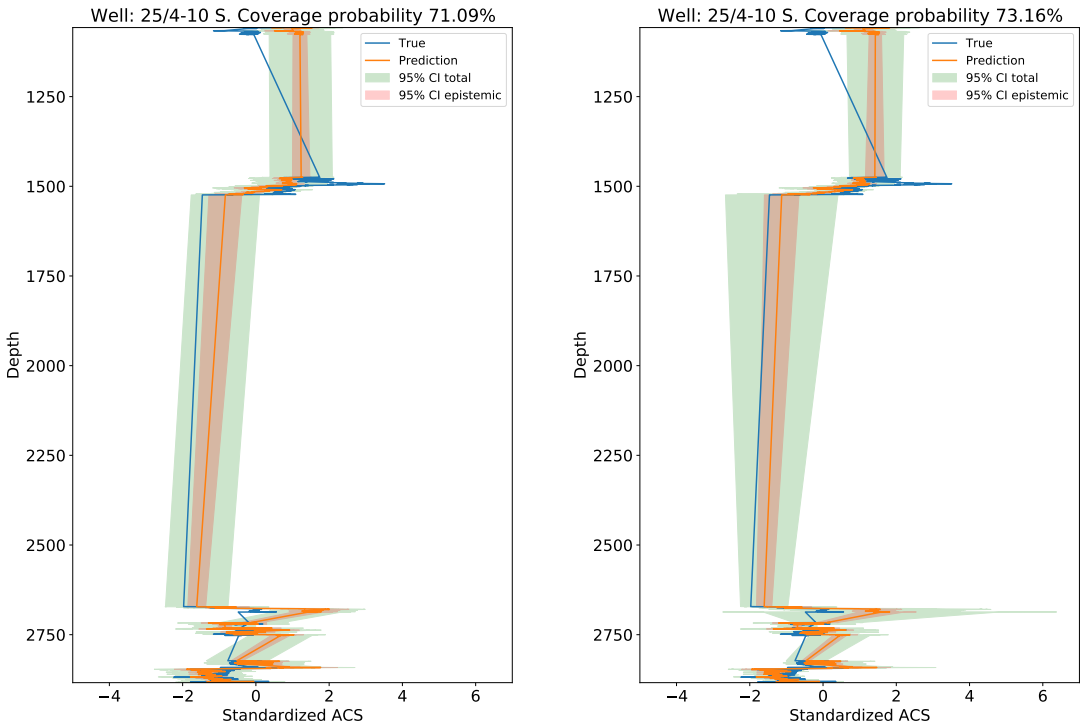


(a) Homoscedastic model

(b) Heteroscedastic model

Figure C.8: Predictions and corresponding 95% credible intervals for well 30/11-11 S in the test set using a (a) homoscedastic and (b) heteroscedastic MC Dropout model. The empirical coverage probability is marked in the title of each plot.

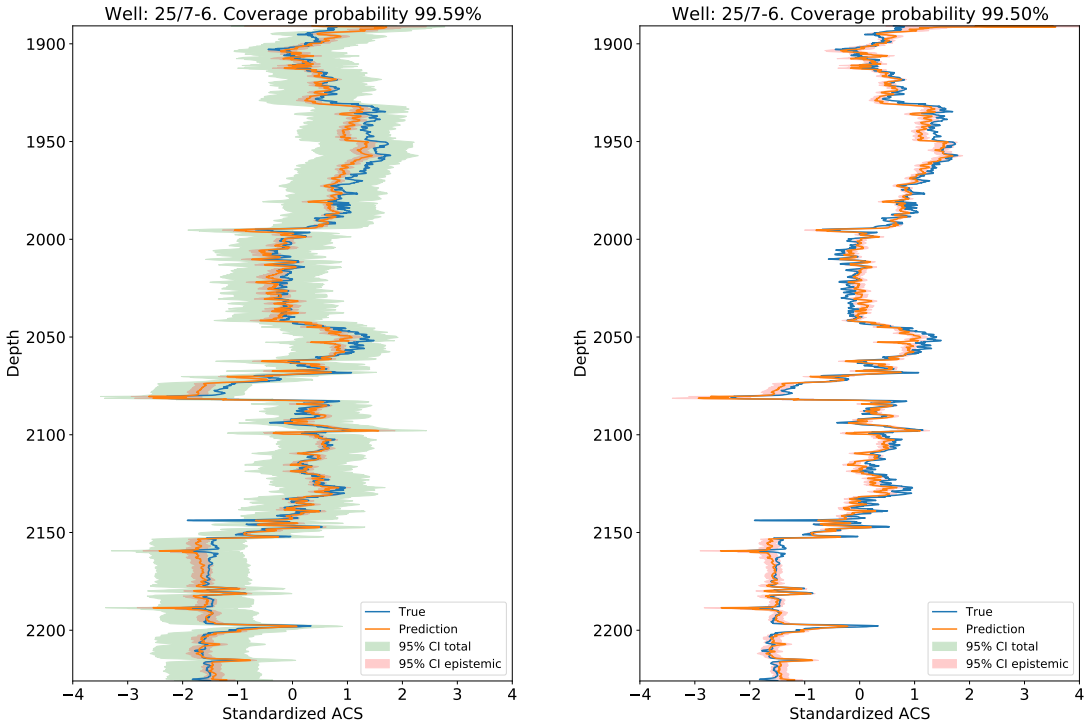
C.2 SGVB



(a) Homoscedastic model

(b) Heteroscedastic model

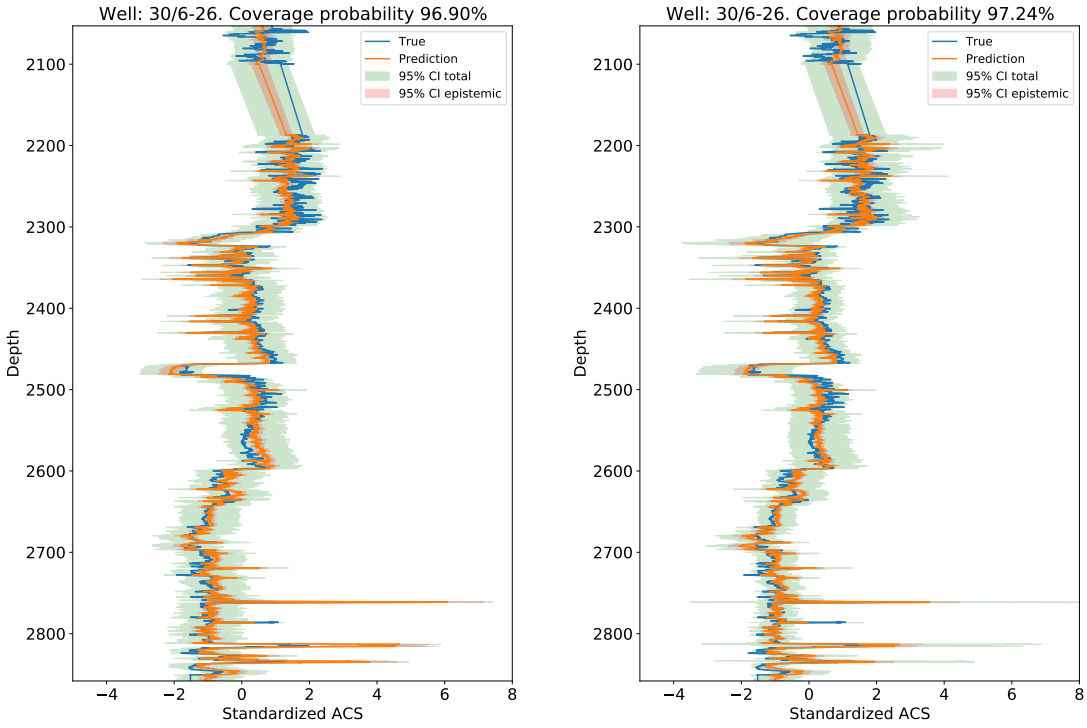
Figure C.9: Predictions and corresponding 95% credible intervals for well 25/4-10 S in the test set using a (a) homoscedastic and (b) heteroscedastic SGVB model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

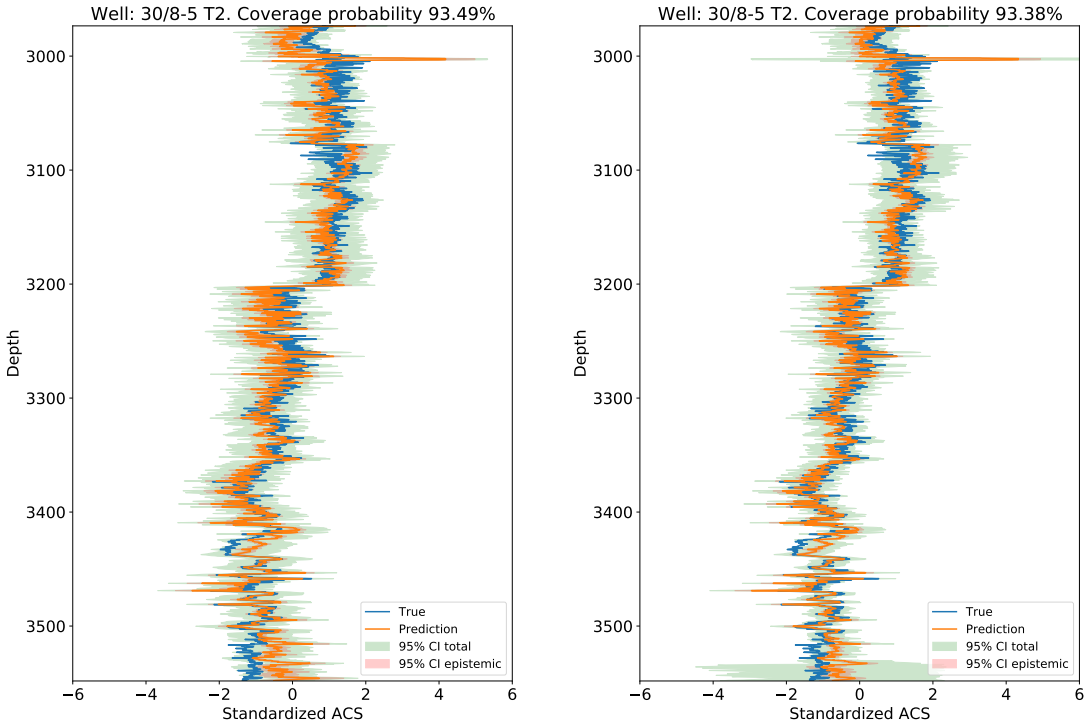
Figure C.10: Predictions and corresponding 95% credible intervals for well 25/7-6 in the test set using a (a) homoscedastic and (b) heteroscedastic SGVB model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

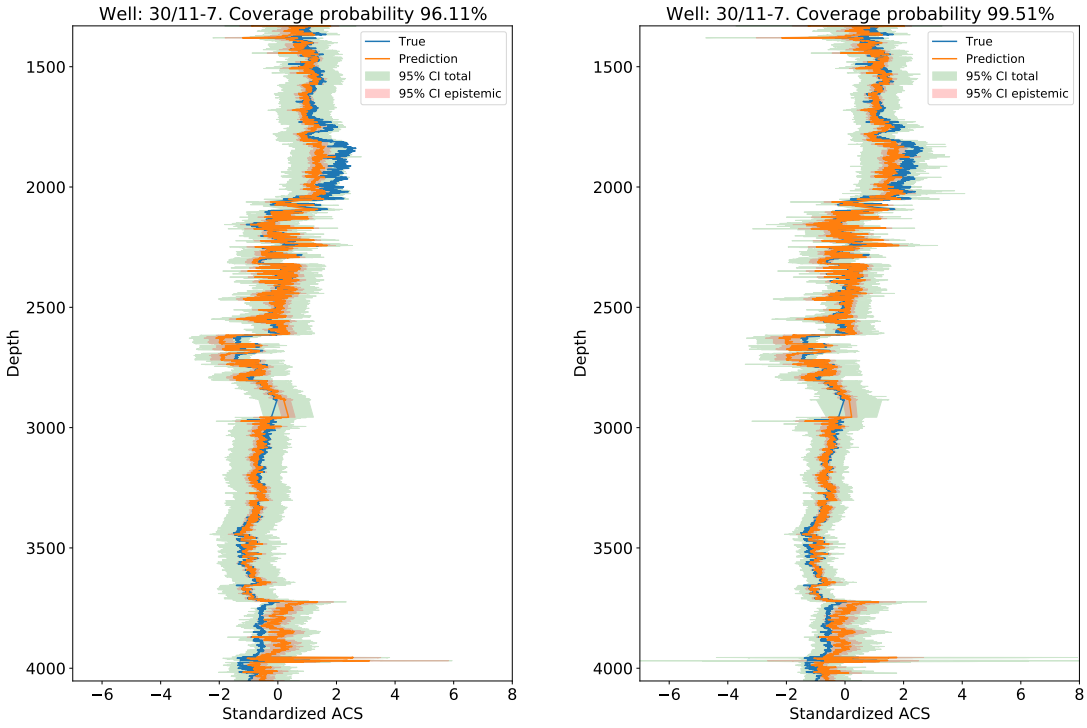
Figure C.11: Predictions and corresponding 95% credible intervals for well 30/6-26 in the test set using a (a) homoscedastic and (b) heteroscedastic SGVB model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

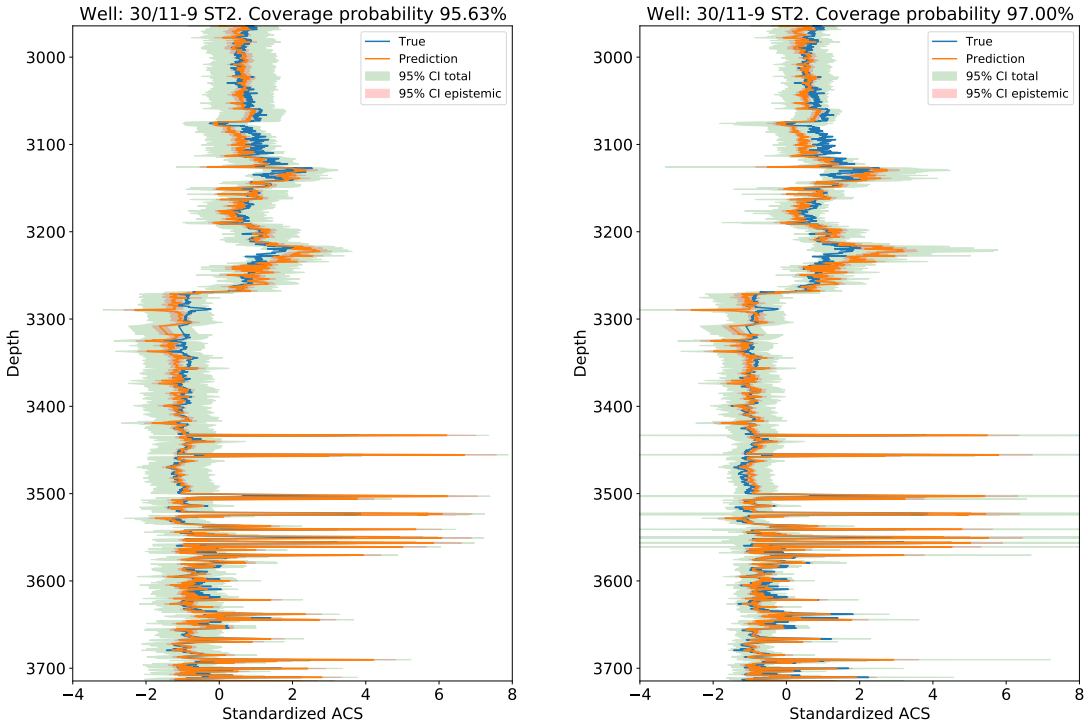
Figure C.12: Predictions and corresponding 95% credible intervals for well 30/8-5 T2 in the test set using a (a) homoscedastic and (b) heteroscedastic SGVB model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

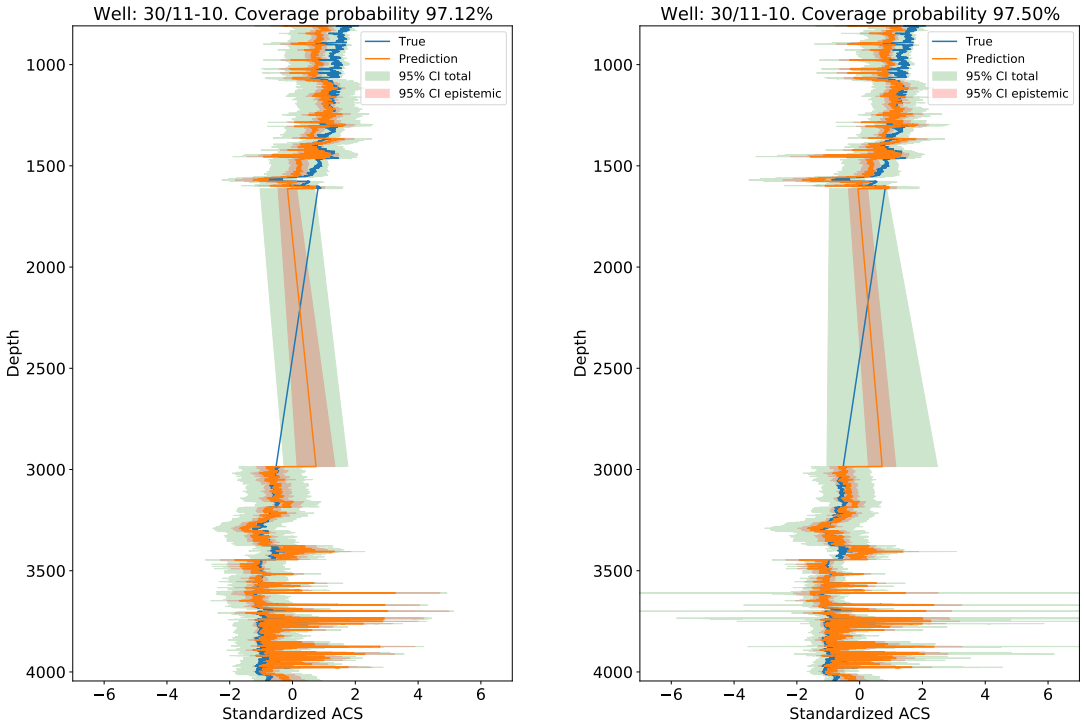
Figure C.13: Predictions and corresponding 95% credible intervals for well 30/11-7 in the test set using a (a) homoscedastic and (b) heteroscedastic SGVB model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

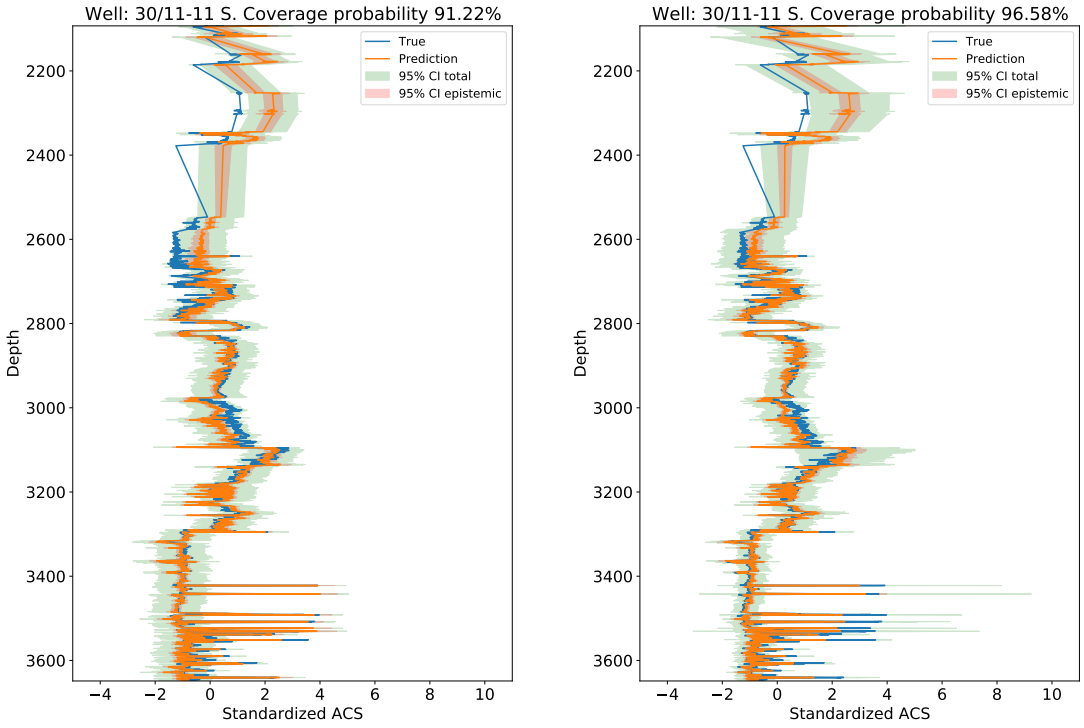
Figure C.14: Predictions and corresponding 95% credible intervals for well 30/11-9 ST2 in the test set using a (a) homoscedastic and (b) heteroscedastic SGVB. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

Figure C.15: Predictions and corresponding 95% credible intervals for well 30/11-10 in the test set using a (a) homoscedastic and (b) heteroscedastic SGVB model. The empirical coverage probability is marked in the title of each plot.



(a) Homoscedastic model

(b) Heteroscedastic model

Figure C.16: Predictions and corresponding 95% credible intervals for well 30/11-11 S in the test set using a (a) homoscedastic and (b) heteroscedastic SGVB model. The empirical coverage probability is marked in the title of each plot.

Appendix D

Calibration curves

In this part of the appendix the calibration curves for all wells are shown for all the considered models. In the first part of the appendix are the calibration curves for the MC Dropout models, while the calibration curves for the SGVB models are shown in the final part. Each part will cover the homoscedastic and heteroscedastic models.

D.1 MC Dropout

D.1.1 Homoscedastic

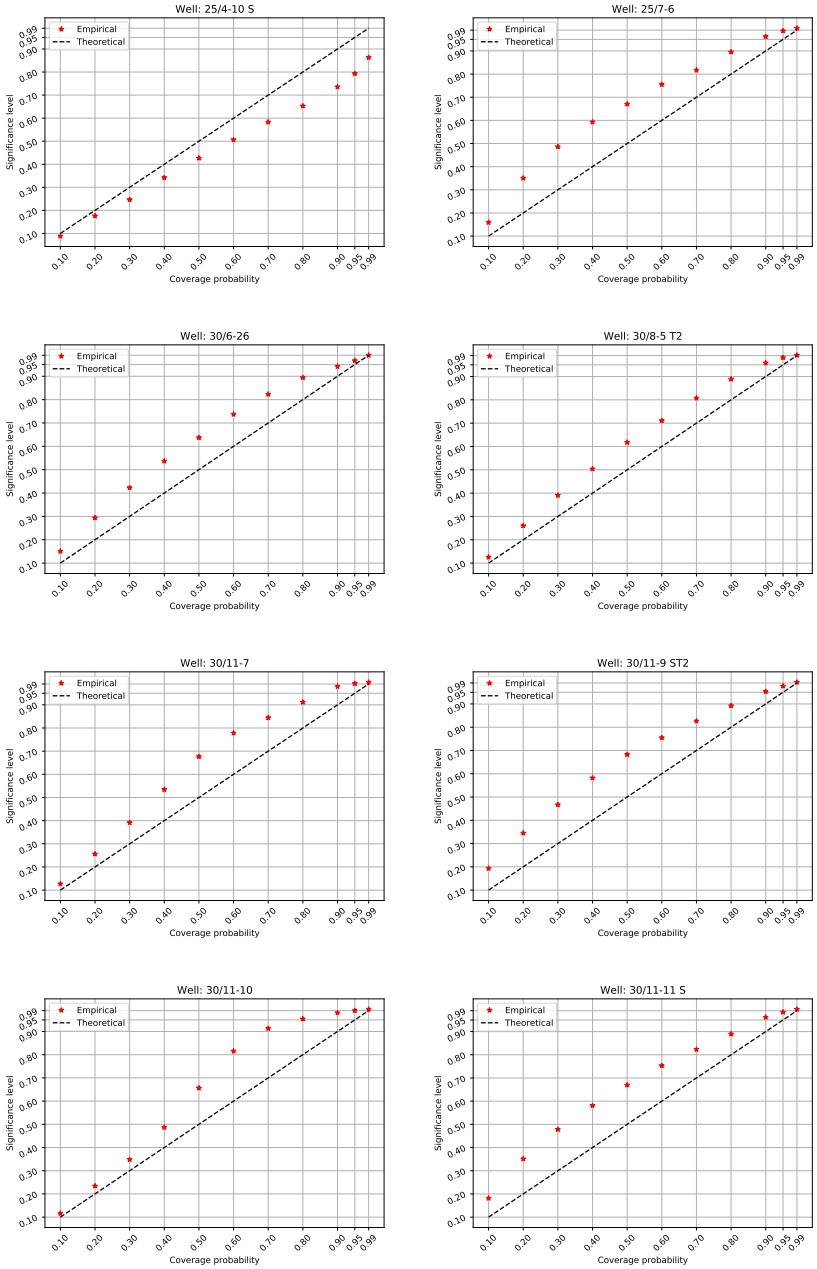


Figure D.1: Calibration curves for all wells in the test set using the homoscedastic MC Dropout model. The name of the well is marked in the title of each figure.

D.1.2 Heteroscedastic

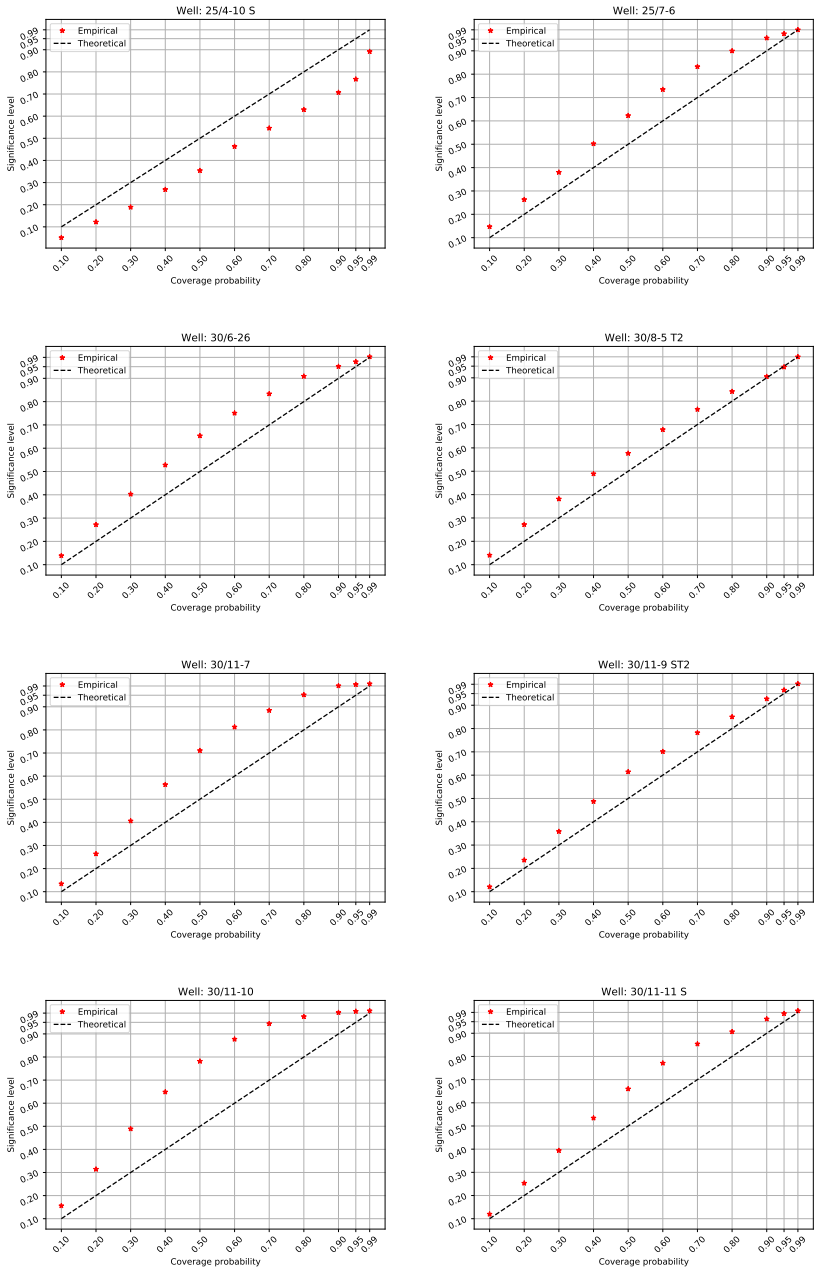


Figure D.2: Calibration plots for all wells in the test set using the heteroscedastic MC Dropout model. The name of the well is marked in the title of each figure.

D.2 SGVB

D.2.1 Homoscedastic

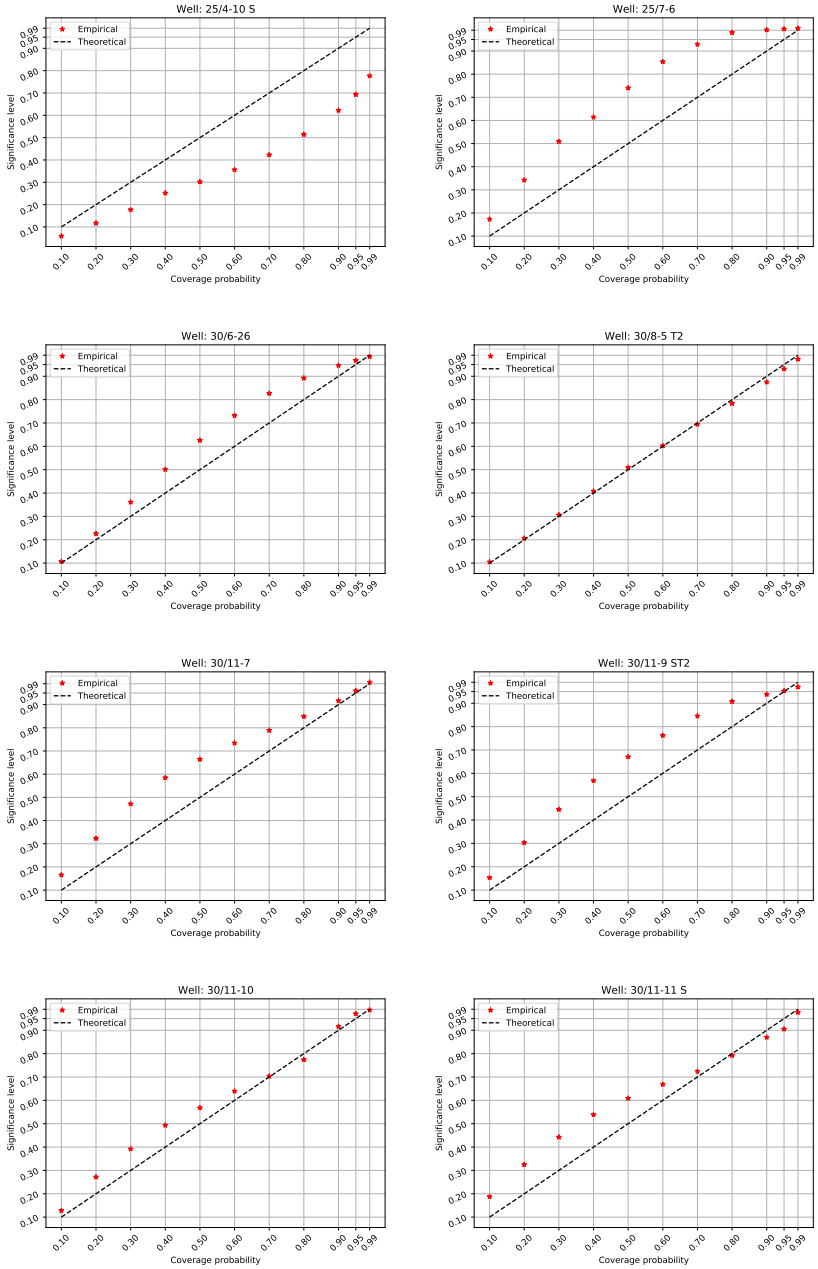


Figure D.3: Calibration plots for all wells in the test set using the homoscedastic SGVB model. The name of the well is marked in the title of each figure.

D.2.2 Heteroscedastic

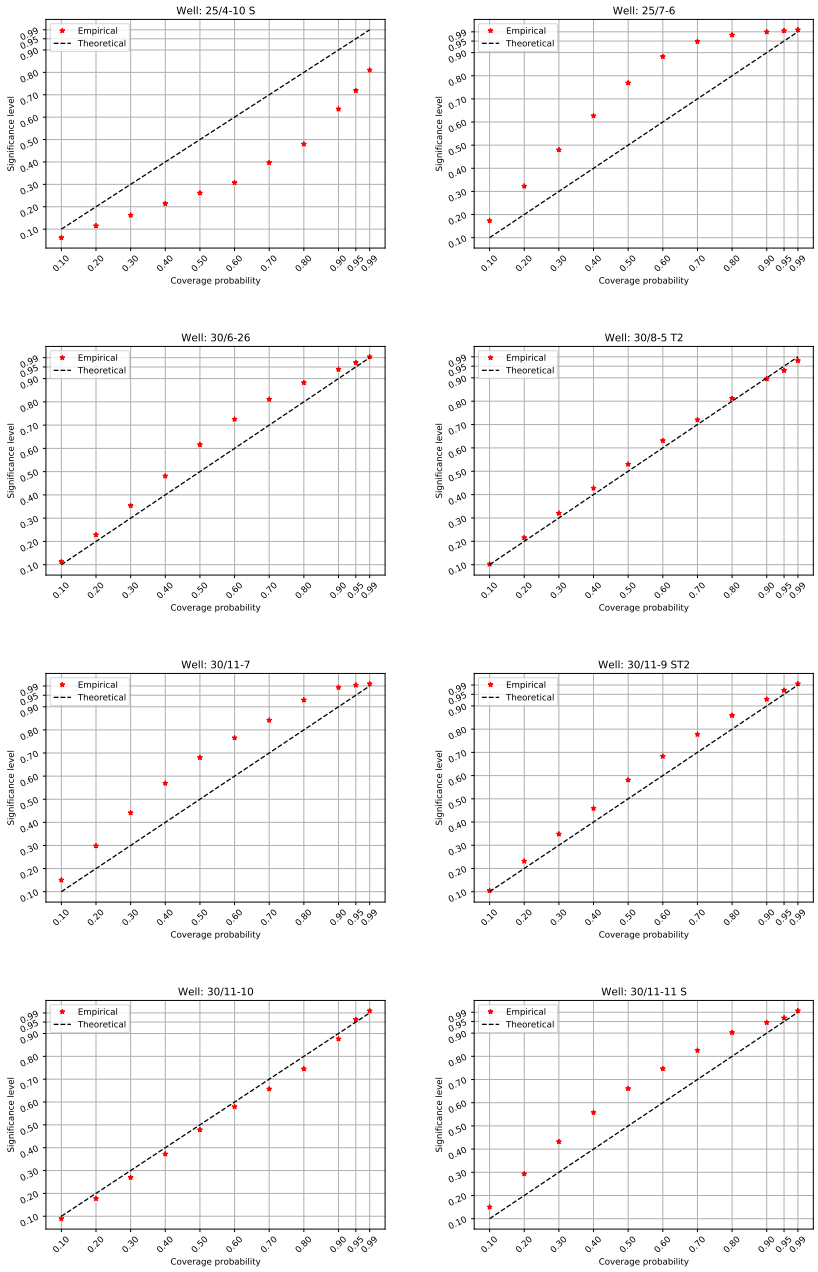


Figure D.4: Calibration plots for all wells in the test set using the heteroscedastic SGVB model. The name of the well is marked in the title of each figure.

Bibliography

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. Rajendra Acharya, V. Makarenkov, and S. Nahavandi. A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges. *arXiv e-prints*, page arXiv:2011.06225, 2020, 2011.06225.

[2] J. Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[3] M. Becquey, M. Lavergne, and C. Willm. Acoustic impedance logs computed from seismic traces. *Geophysics*, 44(9):1485–1501, 1979.

[4] J. O. Berger. *Statistical decision theory and Bayesian analysis; 2nd ed.* Springer Series in Statistics. Springer, New York, 1985. doi:10.1007/978-1-4757-4286-2.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[6] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi:10.1080/01621459.2017.1285773.

[7] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[8] J. Brownlee. Understanding the impact of learning rate on neural network performance. `https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/`. Accessed: 2020-30-11.

[9] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997. doi:https://doi.org/10.1023/A:1007379606734.

[10] A. C. Damianou and N. D. Lawrence. Deep Gaussian Processes. *arXiv e-prints*, page arXiv:1211.0358, 2012, 1211.0358.

[11] J. S. Denker and Y. LeCun. Transforming neural-net output levels to probability distributions. In *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, NIPS'90, page 853–859. Morgan Kaufmann Publishers Inc., 1990.

[12] A. Der Kiureghian and O. Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[13] L. Fahrmeir, T. Kneib, S. Lang, and B. Marx. *Regression: Models, Methods and Applications*. Springer, 2013.

[14] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi:10.1214/aos/1013203451.

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

[16] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Appendix. *arXiv e-prints*, page arXiv:1506.02157, 2015, 1506.02157.

[17] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL `http://www.deeplearningbook.org`.

[19] A. Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

[20] B. Grimstad, M. Hotvedt, A. T. Sandnes, O. Kolbjørnsen, and L. S. Imsland. Bayesian Neural Networks for Virtual Flow Metering: An Empirical Study. *arXiv e-prints*, page arXiv:2102.01391, 2021, 2102.01391.

[21] J. Han, J. Pei, and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.

[22] B. Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *arXiv preprint arXiv:1801.03744*, 2018.

[23] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2008.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[25] G. E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

[26] P. D. Hoff. *A First Course in Bayesian Statistical Methods*. Springer, 2009.

[27] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021. doi:https://doi.org/10.1007/s10994-021-05946-3.

[28] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[29] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[30] A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *arXiv e-prints*, page arXiv:1703.04977, 2017, 1703.04977.

[31] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, 2014, 1412.6980.

[32] D. P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the Local Reparameterization Trick. *arXiv e-prints*, page arXiv:1506.02557, 2015, 1506.02557.

[33] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114, 2013, 1312.6114.

[34] D. P. Kingma and M. Welling. An Introduction to Variational Autoencoders. *arXiv e-prints*, page arXiv:1906.02691, 2019, 1906.02691.

[35] S. Koturwar and S. Merchant. Weight Initialization of Deep Neural Networks(DNNs) using Data Statistics. *arXiv e-prints*, page arXiv:1710.10570, 2017, 1710.10570.

[36] R. Krishnan, M. Subedar, and O. Tickoo. Specifying weight priors in bayesian deep neural networks with empirical bayes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4477–4484, 2020. doi:10.1609/aaai.v34i04.5875.

[37] V. Kuleshov, N. Fenner, and S. Ermon. Accurate Uncertainties for Deep Learning Using Calibrated Regression. *arXiv e-prints*, page arXiv:1807.00263, 2018, 1807.00263.

[38] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL `https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf`.

[39] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015. doi:https://doi.org/10.1038/nature14539.

[40] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. *Neural Networks: Tricks of the Trade: Second Edition*, chapter Efficient BackProp, pages 9–48. Springer, 2012. doi:10.1007/978-3-642-35289-8_3.

[41] D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 1992. doi:10.1162/neco.1992.4.3.448.

[42] T. Mithcell. *Machine Learning*. McGraw Hill, 1997.

[43] R. M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[44] T. O'Hagan. Dicing with the unknown. *Significance*, 2004.

[45] T. Papamarkou, J. Hinkle, M. T. Young, and D. Womble. Challenges in Bayesian inference via Markov chain Monte Carlo for neural networks. *arXiv e-prints*, page arXiv:1910.06539, 2019, 1910.06539.

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv e-prints*, page arXiv:1912.01703, 2019, 1912.01703.

[47] A. T. Sandnes, B. Grimstad, and O. Kolbjørnsen. Multi-task learning for virtual flow metering. *arXiv e-prints*, page arXiv:2103.08713, 2021, 2103.08713.

[48] Schlumberger Oilfield Glossary. Acoustic impedance log. `https://www.glossary.oilfield.slb.com/en/terms/a/acoustic_impedance`. Accessed: 2021-21-04.

[49] Schlumberger Oilfield Glossary. Gamma ray log. `https://www.glossary.oilfield.slb.com/en/terms/g/gamma_ray_log`. Accessed: 2021-16-03.

[50] Schlumberger Oilfield Glossary. Resistivity log. `https://www.glossary.oilfield.slb.com/en/terms/r/resistivity_log`. Accessed: 2021-16-03.

[51] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–549, 2017. URL `http://dx.doi.org/10.1038/nature24270`.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[53] T. J. Sullivan. *Introduction to Uncertainty Quantification*, volume 63. Springer, 2015. doi:10.1007/978-3-319-23395-6.

[54] S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional Variational Bayesian Neural Networks. *arXiv e-prints*, page arXiv:1903.05779, 2019, 1903.05779.

[55] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 270–279, Cham, 2018. Springer International Publishing.

[56] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[57] Tishby, Levin, and Solla. Consistent inference of probabilities in layered networks: predictions and generalizations. In *International 1989 Joint Conference on Neural Networks*, volume 2, pages 403–409, 1989. doi:10.1109/IJCNN.1989.118274.

[58] Tracs International. *Open Hole Wireline Logging - Self Learning Module*.

[59] B.-H. Tran, S. Rossi, D. Milios, and M. Filippone. All You Need is a Good Functional Prior for Bayesian Deep Learning. *arXiv e-prints*, page arXiv:2011.12829, 2020, 2011.12829.

[60] J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal. Uncertainty estimation using a single deep deterministic neural network. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9690–9700. PMLR, 2020. URL `http://proceedings.mlr.press/v119/van-amersfoort20a/van-amersfoort20a.pdf`.

[61] L. Zhu and N. Laptev. Deep and confident prediction for time series at uber. *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017. doi:10.1109/icdmw.2017.19.

Quantifying Predictive Uncertainty in Artificial Neural Networks

NTNU
Norwegian University of
Science and Technology

AkerBP