CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Active Learning and Predictive Modeling Using Uncertainty Quantification

A Bayesian Perspective Applied to Chemical Synthesis Prediction

Master's thesis in Computer science and engineering

Carl Blomgren
Hampus Gummesson Svensson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Active Learning and Predictive Modeling Using Uncertainty Quantification

A Bayesian Perspective Applied to Chemical Synthesis Prediction

Carl Blomgren
Hampus Gummesson Svensson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Active Learning and Predictive Modeling Using Uncertainty Quantification
A Bayesian Perspective Applied to Chemical Synthesis Prediction
Carl Blomgren
Hampus Gummesson Svensson

Supervisor: Yinan Yu, Department of Computer Science and Engineering
Advisor: Simon Johansson, AstraZeneca/Department of Computer Science and Engineering
Examiner: Graham Kemp, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Active Learning and Predictive Modeling Using Uncertainty Quantification
A Bayesian Perspective Applied to Chemical Synthesis Prediction
Carl Blomgren
Hampus Gummesson Svensson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

A deficit with current state-of-the-art machine learning algorithms in drug discovery is that they solely provide a point-estimate. However, in drug discovery, where data is associated with costly and time consuming experiments, there is a need for the models to indicate the uncertainty of their outputs. Otherwise, the models might be used erroneously. In order to obtain uncertainty from the models, this thesis utilizes Bayesian statistical models. In particular, the objective of this thesis is twofold: (1) Investigate the use of uncertainty in active learning (AL) for predicting the observed yields of chemical reactions with different reaction conditions and reactants. Uncertainty methods for AL and methods based on design of experiments were compared. The predictions were done by using the Bayesian probabilistic matrix factorization model Macau. (2) Investigate how the induced uncertainty affects the performance of Bayesian neural networks used to predict reaction conditions. The uncertainty was used to evaluate how reliable the obtained predictions are. The network was based on variational Bayesian methods and we compare *Bayes by Backprop* and *MC dropout* on a severely imbalanced data set. We found that the use of uncertainty in active learning shows better performance with respect to absolute error and variance when a sufficient number of data points have been added to the training set. Also, using uncertainty seems to yield a significant different training set compared to randomly selected points. Bayes by Backprop illustrates comparable accuracy to MC dropout, however, it struggles to predict the minority classes. This further affects the uncertainty estimates on the minority classes which could indicate that MC dropout is more certain than Bayes by Backprop. To conclude, the introduction of uncertainty quantification seems to provide some valuable information to synthesis prediction models. However, future research on the quality of the uncertainty is needed to use the induced uncertainty to its full extent.

Keywords: machine learning, uncertainty quantification, Bayesian probabilistic matrix factorization, Bayesian neural networks, Bayesian statistics, variational inference, active learning, drug discovery, synthesis prediction

Acknowledgements

First and foremost we would like to thank Simon Johansson for our discussions and the valuable input and knowledge that he has provided throughout the project, and Yinan Yu for her insights and support which enabled this project to run smoothly. Also, thank to Graham Kemp for being our examiner. We thank AstraZeneca for providing the resources that were required for this project. In particular, we would like to thank the Molecular AI group at AstraZeneca for providing valuable insights and for welcoming us into the group. It has been truly fascinating to work with such talented individuals. We want to thank Ola Engkvist for his guidance and for the valuable knowledge that he has provided during this project. Lastly, we would like to thank John Daniel Bossér and Erik Sörstadius for their helpful comments on the thesis.

Carl Blomgren and Hampus Gummesson Svensson, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1

Introduction

In recent years, there has been a rise of applications of artificial intelligence (AI) in drug discovery, in particular due to the rapid increase of chemical data over the past decade [1]. The recent advancements in AI, especially in machine learning (ML), have expanded the field of *in silico* experiments to a point where chemical experiments can be conducted by simulation by a computer before they are ever tested experimentally [2]. This is often both cheaper than traditional experiments, and allows for multiple predictions simultaneously, which can aid the chemist in how to plan the experiments [3].

The objective of this thesis is to explore the use of uncertainty to improve in silico experiments for predicting chemical reactions for hit discovery, see Section 1.1. In particular, this thesis investigates the use of uncertainty for expanding a training set of a matrix factorization model and the use of uncertainty as a measure of the quality of the predictions from a neural network.

1.1 The Drug Discovery Process

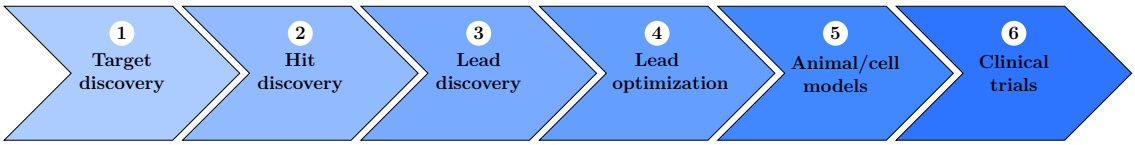


Figure 1.1: The steps of the drug discovery process for obtaining a new drug.

In drug discovery, the first step is to determine a target disease, e.g., COVID-19 or lung cancer. This step also includes determining a physiological drug target, e.g., a specific enzyme or nucleic acid in the body, that is believed to be involved in the disease process [4]. The second step involves selecting appropriate chemical compounds that bind to or modulate some functional signal of the drug target in the body; such that compounds that exceed a threshold value either in binding to the target or modulation of some functional signal are called *hits* [4]. The process of finding hit compounds is called *hit discovery*. During hit discovery, possible hit compounds are selected, usually by experienced scientists, and synthesized. Chemical synthesis is the execution of chemical reactions to obtain one or several products that are part of the reaction. In order to synthesize the possible hit compounds, a chemist plans the reactions in the chemical synthesis that is needed to obtain

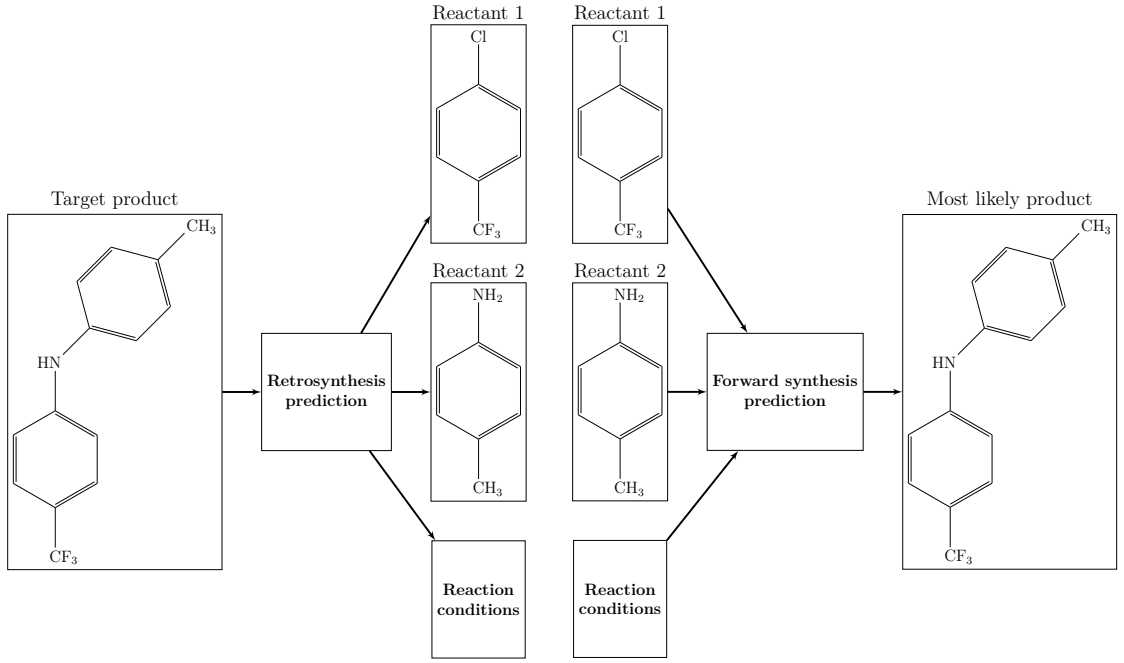
the compound as a product. If the compound is confirmed to be a hit compound in multiple tests and the identity and purity of the compound is confirmed, it will have the status of *validated hit* [4]. Next step is to find a compound or a series of compounds with desirable properties that also fulfills some drug development criteria, e.g., patentability, originality and accessibility by synthesis or extraction [4]. Such a compound is called a *lead*. After this step, the lead is optimized to tune its properties. The resulting optimized lead then becomes a *clinical candidate* if it demonstrates sufficiently low toxicity in cell models and passes animal testing [4]. The clinical candidate is tested on humans to investigate its medical effects more accurately; and the outcome will be a new drug if the clinical candidate is shown to have the desirable effects and a reasonable side-effect profile. The different steps and their order are shown in Figure 1.1. Given the fact that the drug discovery process contains multiple complex steps, it is not a surprise that the entire process, from the initial target discovery to the final product, can take decades and cost in the excess of $1 billion [5].

1.2 AI in Drug Discovery

AI provides promising tools in drug discovery, e.g., where these methods can assist in the prediction of the chemical properties of molecules in a compound [6], [7] (so called molecular property prediction); in the design of new possible hit compounds with sought-after properties [8], [9] (so called *de-novo* design); and in generating feasible sequences of reactions (synthesis) to obtain the compounds [10], [11] (so called synthesis prediction). This thesis focuses on synthesis predictions and, in particular, the subproblem of predicting one reaction (of a sequence).

1.2.1 Synthesis Prediction

There are two categories of synthesis prediction: (1) retrosynthesis and (2) forward synthesis prediction. For a given target compound with a known structure, retrosynthesis prediction means determining valid reactions which could have resulted in this product, as seen in Figure 1.2a [12], [13]. On the other hand, given all available reaction choices, forward synthesis prediction involves deciding which reaction is the most likely to succeed in yielding the desired product. Alternatively, given reactants and reaction conditions, forward synthesis predicts the most likely product of the chemical reaction, as seen in Figure 1.2b [12], [13]. Reaction conditions are defined as the environmental conditions under which the reaction progresses, e.g., temperature, pressure, ligand, solvent and base. A ligand is an ion or molecule that binds to an atom or ion, which is usually metallic, in the center of the molecular structure [14]. A solvent is the substance that the other substances in the reaction are dissolved in. A base is generally a substance that binds to protons (H^+) when dissolved in a solvent. *Reaction components* will be used throughout this thesis to denote both reaction conditions and reactants. Figure 1.3 shows the chemical reaction (with reaction conditions above and below the arrow) that was illustrated in the retrosynthesis and forward synthesis in Figure 1.2. This a reaction from the Merck data, see Section 1.4.1.2. This reaction uses the first aryl halide (as the first



(a) Retrosynthesis prediction **(b)** Forward synthesis prediction

Figure 1.2: Schematics of retrosynthesis and forward synthesis prediction. The target product and reactants are a part of the Merck data used in this thesis, see Section 1.4.1.2.

reactant) while choosing a ligand, an additive and a base is needed in order to predict the yield. The second reactant (not the aryl halide), the solvent DMSO, the temperature 60 °C and the reaction time 16 h are the same for all choices of aryl halide, ligand, additive and base.

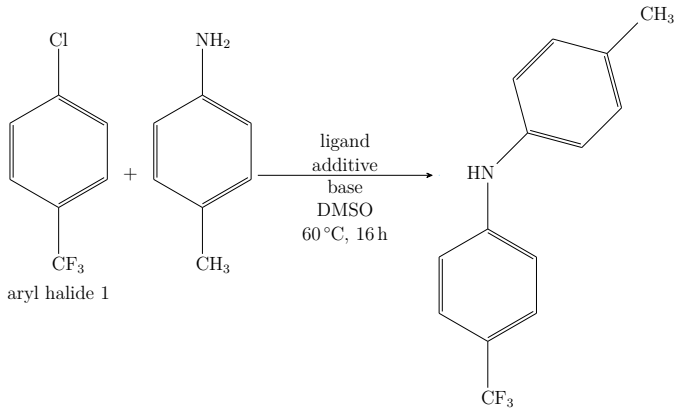


Figure 1.3: A reaction from the Merck data. This reaction uses the first aryl halide while choosing the ligand, additive and base is needed. The second reactant (not the aryl halide), the solvent DMSO, the temperature 60 °C and the reaction time 16 h are the same for all choices of aryl halide, ligand, additive and base.

1.2.2 AI Guided Synthesis Prediction

Synthesis prediction guided by AI has the potential to reduce the time required for finding potential hit compounds by suggesting suitable reactions and conditions, based on models trained on already known data. This is of great interest since the drug discovery process, from target discovery to final product, takes several years [5]. Pharmaceutical companies are trying to make use of the recent technological advances to further automate the drug discovery process, e.g., establishing automated robotic synthesis laboratories. Moreover, this could potentially enable more effective searches of the chemical space and reduce the time required for finding suitable compounds for new drugs, i.e., finding hit compounds [15]. To do this, the performance of the synthesis prediction is essential and, furthermore, it is necessary to prove that it is adequate.

It is also of great importance that the model can indicate the robustness of the predictions. I.e., a model which solely provides a prediction is not sufficient, it is necessary to also quantify the uncertainty in the prediction. This is a consequence of the fact that the required data for synthesis prediction models is associated with real world experiments, making it costly and time consuming to generate data. It is critical to quantify the model's uncertainty when the model is applied to new and more complex data, since otherwise the utilizers of the model may not be aware of its limitations [16]. Furthermore, the estimated uncertainty of the model can be utilized for *Active Learning* (AL), which is when a model guides the acquisition of new data [17]. Utilizing AL has shown useful for chemical prediction models [2], [18]. This is partly due to the associated cost and time required to obtain new data, and partly due to the magnitude of the chemical space, which is estimated to contain more than 10^{60} molecules, which makes it intractable to sample from the entire space [19].

1.3 Objective

The purpose of this thesis is to investigate uncertainty quantification of chemical prediction models relevant to synthesis prediction. In particular, this thesis focuses on exploring the use of uncertainty quantification in AL and as a measure of the quality of the predictions from a neural network. The quality of the predictions are based on the corresponding uncertainty, which can be used to indicate how trustworthy the output from the network is. This could potentially also be used as a metric of where more data is needed. The uncertainty in the neural network is estimated by the use of Bayesian statistical models, which inherently provide a framework for uncertainty quantification. To summarize, this thesis seeks to answer the following questions:

- Can uncertainty successfully be introduced to synthesis prediction models?
- Can uncertainty quantification be successfully applied to AL for synthesis prediction models?
- Can uncertainty quantification be used to evaluate the quality of the predictions of a neural network?

1.4 Delimitations

This thesis focuses on utilizing active learning, based on uncertainty quantification, for extending the training set of the matrix factorization method Macau, which is described in Section 2.5. The goal is to determine the usefulness of such an approach by comparing it with other other approaches for extending the training set. In particular, the aim of the matrix factorization method is to predict the yield from a reaction when using different reaction components, i.e., different reaction conditions and reactants.

Uncertainty quantification in neural networks is based on variational Bayesian methods, the theory for variational methods is presented in Section 2.1.4. The architecture of the networks is limited to feedforward neural networks, which are described in Section 2.4.1. The uncertainty quantification is introduced by utilizing Bayesian neural networks, which are described in Section 2.4.6. This is done in order to investigate if Bayesian feedforward neural networks can be successfully applied to this field. Finally, the aim of the Bayesian network is to predict which ligand that is part of different reactions.

1.4.1 Data Sets

The objective of this thesis is not to investigate predictions on the entire chemical space. Instead, the models in this thesis evaluate predictions on specific chemical data sets, which are relevant for drug discovery. The matrix factorization method utilizes two different data sets of two different reactions that are important to drug discovery. The first data set is published in [20] by the pharmaceutical company Pfizer and it is, therefore, called the Pfizer data. This data set explores a diverse set of reaction components in a Suzuki-Miyaura reaction [21]. Furthermore, the second data set is published in [22] by the pharmaceutical company Merck and it is, therefore, called the Merck data. This data contains experimental results for different reaction components in a Buchwald–Hartwig cross-coupling. These data sets are used since they consist of fully known matrices of combinatorial data, which makes them suitable for a matrix factorization method. The Bayesian neural networks are trained on data from an electronic lab notebook (ELN) obtained from AstraZeneca. Thus, this is proprietary data owned by AstraZeneca and, therefore, not publicly available.

1.4.1.1 Pfizer Data

The Pfizer data [20] consists of four different choices of the first reactant (reactant 1), three different choices of the second reactant (reactant 2), four solvents, eight bases (including one blank option using no base) and twelve ligands (including one blank option using no ligand). These are encoded as a categorical variable for each component. Other reaction components are held constant. Note that in the published data in [20], there are four different available choices of reactant 2. However, the reactant 2 consisting of a Bromide is not used since it utilizes different choices of reactant 1 than the remaining data. For every combination of choices of reaction

components, the data consists of an observed percent yield, which obtains values between 0 and 100, of the product. The product is the main compound formed from the reaction. Therefore, the data can be used as a forward synthesis prediction problem, see Figure 1.2b, if the some or all yields are seen as unknown. The observed yields are given with two decimals. Approximately 3% of all data have an observed yield lower than 5%, which we define as the lowest yield that reaction can obtain to be successful. About 0.3% of all data have an observed yield equal to zero. Note that the observed yield is the true yield that was experimentally measured.

1.4.1.2 Merck Data

The Merck data [22] consists of 16 different choices of aryl halides (including one blank option using no additive), four different choices of ligands, 24 different choices of additives (including one blank options using no additive) and three different choices of bases. See the example in Figure 1.3. Aryl halide is a class of chemical compounds and the different aryl halides in this data are used as one of two reactants. Other reaction components are held constant, including the other reactant. Chemical structures are available for each aryl halide, ligand, additive and base. These chemical structures were used to obtain corresponding 2048-bits ECFP6 fingerprints, see Section 2.7.1.

The choices of reaction components are encoded as a categorical variable for each reaction component. For every combination of choices of reaction components, the data consists of an observed percent yield, which obtains values between 0 and 100, of the product. Therefore, the data can be used as a forward synthesis prediction problem, see Figure 1.2b, if the some or all yields are seen as unknown. Moreover, eight permutations have no observed yield but these missing yields do no restrict this project. The observed yields are given without any decimals. Finally, approximately 25% of all data have an observed yield lower than 5% while about 14% of all points have an observed yield equal to zero.

It should be noted that [23] made comments on the machine learning approach that [22] utilized on the data. [24] is the response to these comments. However, this comment and response is not crucial for this thesis, but rather they bring up thoughts on how to evaluate the performance of machine learning models when exploring chemical data.

1.4.1.3 AstraZeneca ELN

The AstraZeneca (AZ) electronic lab notebook (ELN) data consists of different reactions and the ligand with the greatest yield. In total there are 13 different ligands to choose from. The data is severely imbalanced, where the most frequent ligands account for 54%, 26% and 7% of the total data, respectively. A yield greater than 10% was needed for a reaction to be deemed successful. The input data consists of reaction fingerprints expressed in ECFP6 with 2048-bits, see Section 2.7.2. The output data consists of an array with zeros except for a one at the entry corresponding to the ligand with the greatest yield, i.e., one-hot encoding.

1.5 Thesis Outline

This chapter has given an overview of the drug discovery process, introduced synthesis prediction and its challenges, and presented the data that was explored in this project. The remainder of this document aims to explain the concepts of Bayesian neural networks and matrix factorization, and then present the computational experiments that have been conducted. The goal is to provide enough explanations of the concepts so that readers from different engineering backgrounds can understand the results easily.

Chapter 2 provides an introduction to the theory necessary to understand the methods and results presented in this thesis. In particular, this chapter will first describe computational methods for Bayesian statistics (Section 2.1), followed by a key section which describes how uncertainty can be quantified for Bayesian models (Section 2.2). Thereafter, this chapter presents theory for neural networks in Section 2.4, which is then extended to Bayesian neural networks (Section 2.4.6) used to quantify uncertainty in predictions. Section 2.5 provides an introduction to the matrix factorization method. Lastly, Section 2.7 describes how the molecules and reactions are represented in a computer.

The methods used in this project are presented in Chapter 3. This chapter starts by a description of how matrix factorization is used in active learning to extend a data set. This is followed by a design of experiment approach in Section 3.1.2. Section 3.2 describes how Bayesian neural networks are used to evaluate the uncertainty in predicted reaction conditions. In particular, Section 3.2.1 describes the assumptions made on the variational distribution and how the terms in the loss are evaluated. This is followed by the detailed implementation used on the AZ ELN data (Section 3.2.2), which describe the used hyperparameters of the network. Finally, this chapter concludes with a description of how uncertainty in neural networks are computed.

The results of the conducted computational experiments are shown in Chapter 4. The results of the comparisons between different active learning approaches, and the performance and uncertainty of the Bayesian neural networks are presented. These results are discussed in Chapter 5 where we seek to answer the questions in Section 1.3. Finally, the results and their implications are concluded in Chapter 6.

2

Theory

This chapter aims to provide a brief introduction to the theory that is necessary for this project. Since all models are based on Bayesian statistics Section 2.1 will provide a short introduction to Bayesian statistics and how it can be carried out by a computer. The Bayesian framework provides a natural way for uncertainty quantification, which is described in depth in Section 2.2. This is followed by a brief motivation of why data driven models, such as the Bayesian models that are used in this project, require the data to be partitioned into different subsets (Section 2.3).

This thesis utilizes Bayesian neural networks with a feedforward architecture. In order to understand how these Bayesian networks work, a solid understanding of traditional neural networks and how they are trained is required. Therefore, traditional feedforward neural networks are introduced through Section 2.4.1-2.4.5 and this is then followed by an introduction to Bayesian neural networks in Section 2.4.6.

Moreover, this thesis uses a matrix factorization model when utilizing active learning, based on uncertainty quantification, for predicting the yields from a reaction when using different reaction components. Recall that reaction components denote both the reaction conditions and the reactants. The introduction to traditional feedforward neural networks and Bayesian neural networks is followed by an introduction to matrix factorization (Section 2.5). Probabilistic matrix factorization can be used to estimate the uncertainty in unknown matrix entries.

Furthermore, this thesis uses Kennard-Stone algorithm as an alternative to randomly selecting points. Kennard-Stone algorithm is described in Section 3.1.2. Lastly, Section 2.7 concludes the chapter with a short description of how reactions can be represented in a computer, which is a necessity for any of the models to work.

2.1 Computational Methods for Bayesian Statistics

This section gives an introduction to the computational methods for Bayesian statistics that are used in this thesis. This is relevant since all the applied methods utilizes Bayesian statistics to estimate uncertainty. We start by giving a brief introduction to Bayesian statistics.

2.1.1 Bayesian Statistics

Bayesian statistical methods for inference are based on Bayes' theorem

$$\pi(\boldsymbol{w}|\mathcal{D}) = \frac{\pi(\mathcal{D}|\boldsymbol{w})\pi(\boldsymbol{w})}{\pi(\mathcal{D})}, \tag{2.1}$$

where $\mathcal{D}$ is observed data and $\boldsymbol{w}$ are parameters of a model that might have generated $\mathcal{D}$. Bayes' theorem presents a relationship between the conditional probabilities of $\boldsymbol{w}$ and $\mathcal{D}$ [25]. All the probabilities in Bayes' theorem have different names: $\pi(\boldsymbol{w}|\mathcal{D})$ is the posterior probability, $\pi(\mathcal{D}|\boldsymbol{w})$ is the likelihood function of $\mathcal{D}$, $\pi(\boldsymbol{w})$ is the prior probability and $\pi(\mathcal{D})$ is the marginal likelihood. The posterior gives information of the model parameters $\boldsymbol{w}$ given the observations $\mathcal{D}$. The likelihood expresses how probable the observed data $\mathcal{D}$ is given different parameters $\boldsymbol{w}$ of our model. The prior reflects our prior knowledge of the random variables $\boldsymbol{w}$, since it provides the probability for the variables before any observations have been made. Finally, the marginal likelihood $\pi(\mathcal{D})$ is a normalization constant, such that it guarantees that the posterior distribution is a valid probability distribution.

To do statistical inference using Bayesian statistics, we want to evaluate the posterior probability using Bayes' theorem. Therefore, knowing the marginal likelihood

$$\pi(\mathcal{D}) = \int_{\mathcal{W}} \pi(\mathcal{D}|\boldsymbol{w})\pi(\boldsymbol{w})\mathrm{d}\boldsymbol{w}, \tag{2.2}$$

where $\mathcal{W}$ represents the space of all parameters of the model, is essential in order to perform exact Bayesian inference. Bayesian inference is advantageous, for uncertainty quantification, since it results in a distribution of the sought parameters, rather than optimal point-estimates, which are common in frequentist inference. With the posterior distribution, one can form the posterior predictive distribution

$$\begin{aligned} \pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) &= \mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}\left[\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w})\right] \\ &= \int_{\mathcal{W}} \pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w})\pi(\boldsymbol{w}|\mathcal{D})d\boldsymbol{w}, \end{aligned} \tag{2.3}$$

where $\boldsymbol{y}^*$ denotes the predicted outcome of the model if the input is $\boldsymbol{x}^*$ and the model was trained using the data $\mathcal{D}$. This distribution is useful since it provides insights about the predicted outcome and the corresponding model uncertainty.

Unfortunately, for most complex models the marginal likelihood in Equation (2.2), which is needed to calculate the posterior $\pi(\boldsymbol{w}|\mathcal{D})$, can not be computed analytically, due to the intricate form of the posterior and the vast number of parameters. In fact, exact Bayesian inference is in general intractable due to the integrals in Equation (2.2) and (2.3). However, there are several methods to perform approximate Bayesian inference, where one obtains information from the posterior or probabilities similar to the posterior. The two most common are Markov chain Monte Carlo (MCMC) and Variational Bayes inference (VI). The former method samples from the posterior through different sampling schemes, and the latter approximates the sought posterior $\pi(\boldsymbol{w}|\mathcal{D})$ with a simpler distribution $q(\boldsymbol{w}|\theta)$ which is easier to sample from.

2.1.2 Monte Carlo Integration

In Bayesian statistics one often has to compute complicated multi-dimensional integrals, where a closed form solution generally does not exist [26]. Consider, for example, the integral

$$\int_{\mathcal{W}} g(\boldsymbol{w})\pi(\boldsymbol{w})d\boldsymbol{w},$$

where $\mathcal{W}$ is the parameter space of $\boldsymbol{w}$. This integral can be expressed as the expected value of $g(\boldsymbol{w})$, where $\boldsymbol{w} \sim \pi(\boldsymbol{w})$,

$$\mathbb{E}[g(\boldsymbol{w})] = \mathbb{E}_{\pi(\boldsymbol{w})}[g(\boldsymbol{w})] = \int_{\mathcal{W}} g(\boldsymbol{w})\pi(\boldsymbol{w})d\boldsymbol{w}.$$

Assume that $\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \ldots, \boldsymbol{w}^{(T)}$ are T independent random samples from $\pi(\boldsymbol{w})$, then the average of these samples are

$$\frac{1}{T}\sum_{t=1}^{T} g(\boldsymbol{w}^{(t)}). \tag{2.4}$$

By the *Law of Large Numbers* it follows that the sample average converges to the expected value [27]

$$\frac{1}{T}\sum_{t=1}^{T} g(\boldsymbol{w}^{(t)}) \to \mathbb{E}[g(\boldsymbol{w})], \quad T \to \infty,$$

if $\mathbb{E}[g(\boldsymbol{w}^{(t)})] = \mathbb{E}[g(\boldsymbol{w})]$ and $\mathrm{Var}(g(\boldsymbol{w}^{(t)})) = \mathrm{Var}(g(\boldsymbol{w}))$ and both of these exist, i.e., they are not infinite. This means that the complicated integral can be approximated by the simulation of random numbers. The procedure to generate random numbers and compute their average is generally known as Monte Carlo (MC) integration. Additionally, by the *Central Limit Theorem* the variance of the estimate is given by $\frac{\mathrm{Var}(g(\boldsymbol{w}))}{T}$ [27]. This means that it becomes increasingly less likely that the estimate in Equation (2.4) deviates from the actual expected value when the sample size T increases. Lastly, the assumptions made by Law of Large Numbers also imply that the estimate in Equation (2.4) is unbiased.

2.1.3 Gibbs Sampling

When exact Bayesian inference is not feasible, one possible approach is to use approximate inference methods based on numerical sampling [25]. Such methods often use MCMC algorithms for numerical sampling from a distribution. One popular MCMC algorithm for sampling is Gibbs sampling, which is used to sample from a joint distribution $\pi(x_1, \ldots, x_n)$ of n random variables. In the Gibbs sampler, each variable x_i is sampled from its conditional distribution $\pi(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ sequentially and then the newly sampled value is used when sampling the other subsequent variables. This procedure is shown in Algorithm 1. Note that conditional distributions do not need to be known explicitly as long as it is possible to sample from the distributions. Usually, the outer loop is continued until the joint distribution $\pi(x_1, \ldots, x_n)$ has converged. This procedure samples from the desired distribution

$\pi(x_1, \ldots, x_n)$, since

$$\begin{aligned}\pi(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) &= \frac{\pi(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)}{\pi(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)} \\ &\propto_{x_i} \pi(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n),\end{aligned}$$

when the process has stabilized [28]. The period before the process are stabilized is called burn-in period, where the actual samples from the joint distribution is obtained after this period. The iterations in the burn-in period are called burnins.

Algorithm 1 Gibbs sampling

1: Initialize $\{x_i^1 : i = 1, \ldots, n\}$
2: **for** $\tau = 1, \ldots, T$ **do**
3: **for** $i = 1, \ldots, N$ **do**
4: Sample $x_i^{\tau+1} \sim \pi\left(x_i|x_1^{\tau+1}, \ldots, x_{i-1}^{\tau+1}, x_{i+1}^{\tau}, \ldots, x_n^{\tau}\right)$
5: **end for**
6: **end for**

2.1.4 Variational Bayes

An alternative set of approaches used for approximate Bayesian inference is variational Bayesian methods, also known as variational inference [29]. These can be used to find an approximation of the posterior. The idea is to find a distribution $q(\boldsymbol{w}|\theta)$ with parameters θ, which is easy to evaluate, e.g., a Gaussian, and that is as close as possible to the true posterior $\pi(\boldsymbol{w}|\mathcal{D})$. The approximating distribution $q(\boldsymbol{w}|\theta)$ is known as the *variational posterior distribution* or the *variational distribution*. This distribution can be used to form the variational predictive distribution

$$q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta) = \int_{\mathcal{W}} \pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}) q(\boldsymbol{w}|\theta) d\boldsymbol{w}, \tag{2.5}$$

where $\boldsymbol{y}^*$ denotes the model's prediction of a new input $\boldsymbol{x}^*$, which allows for approximate inference of new data.

The similarity between two distributions can be quantified by the Kullback-Leibler (KL) divergence between the distributions, which is defined as

$$\mathrm{KL}(q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w}|\mathcal{D})) = \int_{\mathcal{W}} q(\boldsymbol{w}|\theta) \log\left(\frac{q(\boldsymbol{w}|\theta)}{\pi(\boldsymbol{w}|\mathcal{D})}\right) \mathrm{d}\boldsymbol{w},$$

where $\pi(\boldsymbol{w}|\mathcal{D}) = 0$ implies that $q(\boldsymbol{w}|\theta) = 0$ for all $\boldsymbol{w}$. Note that $\mathrm{KL}(\pi(\boldsymbol{w}|\mathcal{D})\|q(\boldsymbol{w}|\theta)) \not\equiv \mathrm{KL}(q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w}|\mathcal{D}))$ in general. Furthermore, it holds that $\mathrm{KL}(q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w}|\mathcal{D})) \geq 0$; with equality if and only if $\pi(\boldsymbol{w}|\mathcal{D}) = q(\boldsymbol{w}|\theta)$ [25]. Hence, using KL divergence as a similarity measure, the objective is to find the parameters θ^{opt} that minimizes the KL divergence

$$\theta^{\mathrm{opt}} = \arg\min_{\theta} \ \mathrm{KL}(q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w}|\mathcal{D})).$$

Thus, the variational approach replaces the problem of integration with that of optimization, in order to find the distribution that most resembles the true posterior in a family of distributions.

However, this expression of the KL divergence does not lend itself to direct minimization, since it contains the unknown posterior $\pi(\boldsymbol{w}|\mathcal{D})$. This expression can be rephrased in an equivalent form which does not depend directly on the posterior,

$$\begin{aligned}
\theta^{\text{opt}} &= \arg\min_{\theta} \text{KL}[q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w}|\mathcal{D})] \\
&= \arg\min_{\theta} \int_{\mathcal{W}} q(\boldsymbol{w}|\theta) \log\left(\frac{q(\boldsymbol{w}|\theta)}{\pi(\boldsymbol{w}|\mathcal{D})}\right) d\boldsymbol{w} \\
&= \arg\min_{\theta} \int_{\mathcal{W}} q(\boldsymbol{w}|\theta) \log\left(\frac{q(\boldsymbol{w}|\theta)\pi(\mathcal{D})}{\pi(\mathcal{D}|\boldsymbol{w})\pi(\boldsymbol{w})}\right) d\boldsymbol{w} \\
&= \arg\min_{\theta} \Bigg(\int_{\mathcal{W}} q(\boldsymbol{w}|\theta) \log\left(\frac{q(\boldsymbol{w}|\theta)}{\pi(\boldsymbol{w})}\right) d\boldsymbol{w} \\
&\quad - \int_{\mathcal{W}} q(\boldsymbol{w}|\theta) \log\left(\pi(\mathcal{D}|\boldsymbol{w})\right) d\boldsymbol{w} + \int_{\mathcal{W}} q(\boldsymbol{w}|\theta) \log\left(\pi(\mathcal{D})\right) d\boldsymbol{w} \Bigg) \\
&= \arg\min_{\theta} \Big(\text{KL}[q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log\left(\pi(\mathcal{D}|\boldsymbol{w})\right)] + \log\left(\pi(\mathcal{D})\right) \Big) \\
&= \arg\min_{\theta} \Big(\text{KL}[q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log\left(\pi(\mathcal{D}|\boldsymbol{w})\right)] \Big).
\end{aligned} \tag{2.6}$$

The third equality follows from Bayes theorem, Equation (2.1), and the fourth is a consequence of the fact that integration is a linear operator. Finally, since $\log(\pi(\mathcal{D}))$ is independent of θ it can be omitted from the objective, explaining the last equality. The obtained loss function in Equation (2.6) is known as the *variational free energy* [30]. Lets denote this loss

$$\mathcal{F}(\mathcal{D}, \theta) := \text{KL}[q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log\left(\pi(\mathcal{D}|\boldsymbol{w})\right)]. \tag{2.7}$$

The loss consists of two terms. The first term depends solely on the prior $\pi(\boldsymbol{w})$ and the variational posterior, and not directly on the data. This term punishes the complexity of the posterior and thereby acts as a regularizer of the model. The second term depends on the data and favors the posterior's ability to explain the data. The first and second term will be referred to as the *complexity loss* and *likelihood loss*, respectively. In general $\mathcal{F}(\mathcal{D}, \theta)$ is intractable, however using the definition of KL implies that $\mathcal{F}(\mathcal{D}, \theta)$ can be expressed as

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

which makes it clear that it can be approximated by T number of MC samples

$$\mathcal{F}(\mathcal{D}, \theta) \approx \frac{1}{T}\sum_{i=1}^{T} \log\left(\frac{q(\boldsymbol{w}^{(i)}|\theta)}{\pi(\boldsymbol{w}^{(i)})}\right) - \log\left(\pi(\mathcal{D}|\boldsymbol{w}^{(i)})\right), \quad \boldsymbol{w}^{(i)} \sim q(\boldsymbol{w}|\theta). \tag{2.8}$$

2.2 Uncertainty Quantification

This section introduces uncertainty quantification, which is used to estimate the uncertainty in both the matrix factorization method and the Bayesian neural network. The decomposition presented here is limited to classification, and therefore

is only used by the neural network to quantify the uncertainty in predicted reaction conditions.

There are two main categories of uncertainties in a Bayesian model: *aleatoric* and *epistemic* [26], [31]. These two uncertainties can be used to indicate how certain a model is with its predictions. Aleatoric uncertainty is a measure of the inherent noise in the data. This means that the aleatoric uncertainty cannot be reduced by obtaining more data or by modifying the model. An example of aleatoric uncertainty is the measurement imprecision of the equipment. Epistemic uncertainty originates due to simplifications in the assumptions of the model, i.e., uncertainty in the model parameters. This uncertainty can thereby be reduced given enough data since then the model parameters can be adjusted to better represent the data. Thus it is of great interest to differentiate between aleatoric and epistemic uncertainty, since the latter indicates how much a model can improve whereas the former will not improve unless the data acquisition is improved [26], [31], [32]. Aleatoric uncertainty can be further divided into *homoscedastic* and *heteroscedastic* aleatoric uncertainty. The former stays constant over different input patterns, whereas the latter varies with the input [31], [26].

Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i$ is the i-th input and $\boldsymbol{y}_i$ is the corresponding output, be the data consisting of input patterns that the Bayesian model has been trained on and N is the number of input patterns in the data. A possible way to estimate the uncertainty in classification, for a new input $\boldsymbol{x}^*$ and the corresponding output $\boldsymbol{y}^*$, is to compute the entropy of the posterior predictive

$$H\left(\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D})\right) = -\sum_{k}^{K} \pi(\boldsymbol{y}^* = e_k|\boldsymbol{x}^*, \mathcal{D}) \log\left(\pi(\boldsymbol{y}^* = e_k|\boldsymbol{x}^*, \mathcal{D})\right),$$

where e_k is a one-hot encoded vector with 1 at element k and K is the total number of classes [26]. The entropy is a measure of the averaged information available in the distribution, which can be interpreted as how uncertain the distribution is. However, the posterior predictive $\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D})$ depends on the true posterior, which generally does not have a closed-form expression. Therefore, in variational inference the posterior predictive is generally replaced with the variational predictive $q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta)$, Equation (2.5), which has the entropy

$$H\left(q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta)\right) = -\sum_{k}^{K} q(\boldsymbol{y}^* = e_k|\boldsymbol{x}^*, \theta) \log\left(q(\boldsymbol{y}^* = e_k|\boldsymbol{x}^*, \theta)\right).$$

For a general model likelihood $\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w})$ there is no analytical expression for $q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta)$. However, the integral can be approximated with MC integration

$$q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta) \approx \overline{p}(\boldsymbol{x}^*) = \frac{1}{T}\sum_{t=1}^{T} \hat{p}^{(t)}(\boldsymbol{x}^*) \quad \boldsymbol{w}^{(t)} \sim q(\boldsymbol{w}|\theta), \tag{2.9}$$

where $\hat{p}^{(t)}(\boldsymbol{x}^*)$ denotes a realization of the model likelihood $\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}^{(t)})$ for a model with parameters $\boldsymbol{w}^{(t)}$. This yields the approximation

$$H\left(q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta)\right) \approx -\sum_{k}^{K} \overline{p}(\boldsymbol{y}^* = e_k|\boldsymbol{x}^*) \log\left(\overline{p}(\boldsymbol{y}^* = e_k|\boldsymbol{x}^*)\right) \tag{2.10}$$

for the entropy of the variational predictive [26]. A deficit of this uncertainty estimate is that it does not differentiate between aleatoric and epistemic uncertainty.

The variance of the posterior predictive distribution can also be used to quantify the uncertainty of a new output $\boldsymbol{y}^*$

$$\mathrm{Var}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}(\boldsymbol{y}^*) = \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*\boldsymbol{y}^{*T}] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]^T.$$

Note that $\mathrm{Var}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}(\boldsymbol{y}^*)$ is the variance of a random vector, thereby it will yield the covariance matrix for $\boldsymbol{y}^*$ under $\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})$. By utilizing the law of total variance, [33] decomposed this variance into aleatoric and epistemic uncertainty for classification problems

$$\begin{aligned}\mathrm{Var}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}(\boldsymbol{y}^*) = \mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}\Bigg[&\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\right) \\ &\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\right)^T\Bigg] \\ + \mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}&\left[\mathrm{diag}\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]\right) - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]^T\right],\end{aligned} \tag{2.11}$$

where $\mathrm{diag}(v)$ is a diagonal matrix with the elements of vector v and $\boldsymbol{w}$ is the parameter vector. The first and second term are the epistemic and aleatoric uncertainty, respectively [33]. Finally, note that it is critical that $\boldsymbol{y}^*$ is one-hot encoded for this decomposition to be valid.

To understand why these terms represent the aleatoric and epistemic uncertainty note that

$$\begin{aligned}&\mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}\left[\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\right] = \\ &\int_{\mathcal{W}}\int_{\mathcal{Y}} \boldsymbol{y}^*\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})d\boldsymbol{y}\pi(\boldsymbol{w}|\mathcal{D})d\boldsymbol{w} - \int_{\mathcal{W}}\int_{\mathcal{Y}} \boldsymbol{y}^*\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})d\boldsymbol{y}\pi(\boldsymbol{w}|\mathcal{D})d\boldsymbol{w} = \\ &\int_{\mathcal{Y}} \boldsymbol{y}^*\int_{\mathcal{W}} \pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})\pi(\boldsymbol{w}|\mathcal{D})d\boldsymbol{w}d\boldsymbol{y} - \int_{\mathcal{Y}} \boldsymbol{y}^*\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})d\boldsymbol{y}\int_{\mathcal{W}} \pi(\boldsymbol{w}|\mathcal{D})d\boldsymbol{w} = \\ &\int_{\mathcal{Y}} \boldsymbol{y}^*\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})d\boldsymbol{y} - \int_{\mathcal{Y}} \boldsymbol{y}^*\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})d\boldsymbol{y} = \boldsymbol{0},\end{aligned}$$

where the second to last equality follows from the definition of the posterior predictive, Equation (2.3). Utilizing this result yields that Equation (2.11) can be expressed as

$$\begin{aligned}\mathrm{Var}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}(\boldsymbol{y}^*) = \mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}\Bigg[&\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\right) \\ &\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\right)^T\Bigg] \\ - \mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}&\left[\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\right] \\ &\mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}\left[\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\right]^T \\ + \mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}&\left[\mathrm{diag}\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]\right) - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]^T\right] \\ = \mathrm{Var}_{\pi(\boldsymbol{w}|\mathcal{D})}&\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\mathcal{D})}[\boldsymbol{y}^*]\right) \\ + \mathbb{E}_{\pi(\boldsymbol{w}|\mathcal{D})}&\left[\mathrm{diag}\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]\right) - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]^T\right].\end{aligned}$$

The first and second terms in this expression are the epistemic and aleatoric uncertainty, respectively. The first term measures the variability of the output due to the randomness of $\boldsymbol{w}$ given the data $\mathcal{D}$. This interpretation can be explained by the fact that it contains a difference between the expected outputs over the likelihood $\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w})$ and posterior predictive $\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D})$. This basically means that it measures the variance of how much the output deviates from the posterior predictive of the model, i.e., the epistemic uncertainty. This quantity can be reduced given more data [33]. The second term, on the other hand, measures the expected variance in the output, w.r.t. the current model. Thereby this term measures the variance in the data, i.e., the aleatoric uncertainty.

A problem with the decomposition in Equation (2.11) is that it utilizes the true posterior. As previously mentioned, the posterior does not generally have a closed-form expression, which thereby does not allow for exact Bayesian inference. However, [33] proposed an approach where a variational distribution is used instead of the true posterior. In variational inference, the posterior $\pi(\boldsymbol{w}|\mathcal{D})$ and posterior predictive $\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D})$ in Equation (2.11) are replaced by the variational distribution $q(\boldsymbol{w}|\theta)$ and the variational predictive $q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta)$, respectively, so that this expression takes the form

$$\begin{aligned}\mathrm{Var}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}(\boldsymbol{y}^*) = \mathbb{E}_{q(\boldsymbol{w}|\theta)}\Bigg[&\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*]\right) \\ &\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] - \mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*]\right)^T \Bigg] \\ + \mathbb{E}_{q(\boldsymbol{w}|\theta)}&\left[\mathrm{diag}\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]\right) - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*]^T\right],\end{aligned} \tag{2.12}$$

where $q(\boldsymbol{w}|\theta)$ is the variational distribution, parameterized by θ, and $q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta)$ is the variational predictive, Equation (2.5) [33]. An unbiased estimate of Equation (2.12) can be obtained by MC integration

$$\begin{aligned}\mathrm{Var}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}(\boldsymbol{y}^*) \approx \frac{1}{T}\sum_{t=1}^{T} &\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*] - \mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*]\right) \\ &\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*] - \mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*]\right)^T \\ + \frac{1}{T}\sum_{t=1}^{T}&\left[\mathrm{diag}\left(\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*]\right) - \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*]\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*]^T\right],\end{aligned} \tag{2.13}$$

where $\boldsymbol{w}^{(t)}$ are the t-th sample from $q(\boldsymbol{w}|\theta)$ and T is the number of samples. One can thereby obtain an unbiased estimate of the variational predictive if the remaining expectations can be computed.

Note that for classification problems $\boldsymbol{y}^*$ is usually one-hot encoded, i.e., only one of its elements is equal to one while the others are equal to zero. Furthermore, since the expectation of a random vector is the expectation of each element, the elements of the expectation $\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*]$ will be the probability that the element is equal to one under the model likelihood with weight sample $\boldsymbol{w}^{(t)}$. The expectation $\mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}$ will analogously be the probability that the corresponding element is equal to one

under the variational predictive distribution. For example, if there are only two possible classes the expectations in Equation (2.13) evaluate to

$$\begin{aligned}\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[\boldsymbol{y}^*] &= \begin{bmatrix}\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[y_1^*] \\ \mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w})}[y_2^*]\end{bmatrix} = \begin{bmatrix}\pi((1,0)^T|\boldsymbol{x}^*,\boldsymbol{w})\cdot 1 + \pi((0,1)^T|\boldsymbol{x}^*,\boldsymbol{w})\cdot 0 \\ \pi((1,0)^T|\boldsymbol{x}^*,\boldsymbol{w})\cdot 0 + \pi((0,1)^T|\boldsymbol{x}^*,\boldsymbol{w})\cdot 1\end{bmatrix} \\ &= \begin{bmatrix}\pi((1,0)^T|\boldsymbol{x}^*,\boldsymbol{w}) \\ \pi((0,1)^T|\boldsymbol{x}^*,\boldsymbol{w})\end{bmatrix},\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*] &= \begin{bmatrix}\mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[y_1^*] \\ \mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[y_2^*]\end{bmatrix} = \begin{bmatrix}q((1,0)^T|\boldsymbol{x}^*,\theta)\cdot 1 + q((0,1)^T|\boldsymbol{x}^*,\theta)\cdot 0 \\ q((1,0)^T|\boldsymbol{x}^*,\theta)\cdot 0 + q((0,1)^T|\boldsymbol{x}^*,\theta)\cdot 1\end{bmatrix} \\ &= \begin{bmatrix}q((1,0)^T|\boldsymbol{x}^*,\theta) \\ q((0,1)^T|\boldsymbol{x}^*,\theta)\end{bmatrix}.\end{aligned}$$

The case with K different classes is obviously analogous, such that each element will be the probability that the corresponding element is equal to one.

Let $\hat{p}^{(t)}(\boldsymbol{x}^*)$ denote the output, for MC sample t, from a model with parameters $\boldsymbol{w}^{(t)}$ and input $\boldsymbol{x}^*$. Furthermore, assume that the output $\hat{p}^{(t)}(\boldsymbol{x}^*)$ from the model can be interpreted as probabilities. In particular, assume that each element of $\hat{p}^{(t)}(\boldsymbol{x}^*)$ corresponds to the probability that the model would predict the corresponding class. Under these assumptions one can approximate $\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*]$ with [33]

$$\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*] \approx \hat{p}^{(t)}(\boldsymbol{x}^*).$$

The expectation $\mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*]$ depends on the variational predictive distribution $q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)$, Equation (2.5). The MC approximation of this distribution was presented in Equation (2.9). This yields the approximation [33]

$$\mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*] \approx \overline{p}(\boldsymbol{x}^*) = \frac{1}{T}\sum_{t=1}^{T}\hat{p}^{(t)}(\boldsymbol{x}^*),$$

which again only depends on the output from the model. Under these approximations Equation (2.13) takes the form

$$\begin{aligned}\mathrm{Var}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}(\boldsymbol{y}^*) \approx &\frac{1}{T}\sum_{t=1}^{T}\left(\hat{p}^{(t)}(\boldsymbol{x}^*) - \overline{p}(\boldsymbol{x}^*)\right)\left(\hat{p}^{(t)}(\boldsymbol{x}^*) - \overline{p}(\boldsymbol{x}^*)\right)^T \\ &+ \frac{1}{T}\sum_{t=1}^{T}\left[\mathrm{diag}\left(\hat{p}^{(t)}(\boldsymbol{x}^*)\right) - \hat{p}^{(t)}(\boldsymbol{x}^*)\hat{p}^{(t)}(\boldsymbol{x}^*)^T\right].\end{aligned} \tag{2.14}$$

It is worth noting that Kendall and Gal suggested an alternative way of quantifying the aleatoric and epistemic uncertainty for neural networks [31]. Their approach introduces extra nodes for the mean and variance in the final layer. Thereby doubling the number of nodes in the final layer compared to networks where the uncertainty is not desired or networks that utilizes the decomposition in Equation (2.11). Furthermore, this approach for uncertainty quantification has some additional deficiencies for classification problems, where the most prominent is that it does not model the variance in the predictive probability [33].

2.3 Training, Testing and Validating a Model

This section provides a short motivation to why it is necessary to split the data into different subsets when using data-driven models. This partition of the data is used by all models presented in this thesis.

When working with data-driven models and algorithms, such as supervised Machine Learning, one usually splits the data into three subsets: a training set, a validation set and a test set. The idea is that each of these subsets shall be representative of the population from which the data is acquired, and that the sets are independent of each other. I.e. the subsets are assumed to be from the same distribution, but where the noise inherent to the observation does not generalize between the sets.

The training data consists of samples which are used to train the model, i.e., fit the model parameters so that the model captures the inherent features of the data. In a supervised setting this means that the model output is trained to better coincide with the observed outcome. However, a problem is that data does not solely consist of an underlying truth, it also has some noise. A problem of data driven methods is therefore to distinguish between the actual characteristics of the data which it is trying to learn, from the noise inherent to all observations. If the model starts to learn features that are specific for the training data then the model is said to *overfit* the training data. This means that it learns noise from the training data and not features that generalize to other data from the same population. A consequence of this is that it usually results in worse performance on the data which the model is not training on. Thereby, it is not desirable for the model to overfit the training data.

This problem can be addressed by the use of a validation set. This is a set from the same distribution as the training set, thereby it contains the same sought characteristics, but it consists of data which is not incorporated into the training set. The idea is that this set can be used to monitor the model performance, whilst not providing any feedback to the model itself. This last part is essential since it means that the model does not learn from the validation data. This set can instead be used to decide when the model starts to learn non-generalizable features, thereby indicating when the training of the model should stop. Figure 2.1 illustrates this idea.

The procedure of utilizing a validation set to indicate when the model is finished with training does however bias the model towards the validation set. Therefore, a test set, consisting of previously unseen data, is used to obtain an unbiased estimate of the model's performance. In general, one wants the performance on the different data sets to be similar, since this indicates that the model has learnt actual characteristics of the data. An additional benefit of these three splits is that hyperparameter-tuning, which is present in most data driven models, can be carried out on the validation data. This further increases the model bias towards the validation data, since the model indirectly learns from this data. But the additional test set can still be used to indicate whether the model performance is generalizable or not.

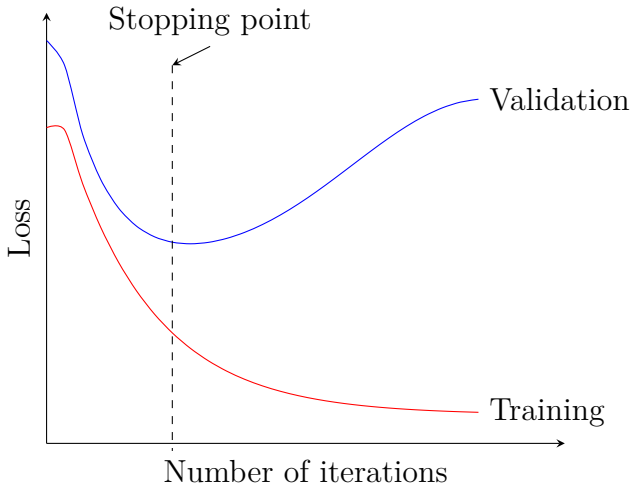


Figure 2.1: The idea of how to monitor the model performance using a validation set. One should stop training the model when the validation loss starts to increase, since then the model starts to overfit the training data.

A potential problem is that the model's performance might depend on how the data is partitioned, i.e., it depends on how the observations are distributed among the different sets. A potential approach that can be used to reduce the impact of this partitioning is K-fold Cross-Validation (CV). The data is then split into K folds, and the model is trained on $K-1$ folds and the K-th fold is the validation set, thereafter the next fold is used as validation data and the remaining are training and so on. Algorithm 2 illustrates the idea of CV.

Algorithm 2 K-fold Cross validation

```
Pick K ∈ ℕ
for k = 1, ..., K do
    train_fold_k, validation_fold_k = cv_split(k, K, data)
    model = fit(train_fold_k)
    performance[k] = evaluate(model, validation_fold_k)
end for
```

2.4 Neural Networks

This section seeks to provide an introduction to feedforward neural networks and how they are trained. Firstly, it covers the theory of traditional feedforward networks and how they are trained. Secondly, it introduces Bayesian neural networks. Bayesian neural networks have been used to quantify the networks' uncertainty in the predicted reaction conditions.

The term artificial neural network (ANN), or just simply denoted neural network (NN), is a widely used expression that covers a large extent of learning models and methods. These neural networks are inspired by the biological neural networks, e.g.,

the neural network in our brains that contains millions of neurons which are connected by synapses [34]. An artificial neural network consists of artificial neurons, hereafter only denoted as neurons, which are able to transfer information between each other. Usually, the neurons are placed in layers, where neurons transfer information between the layers, see Figure 2.2. To the scope of this thesis, we hereafter focus on introducing feedforward neural networks.

2.4.1 Feedforward Neural Network

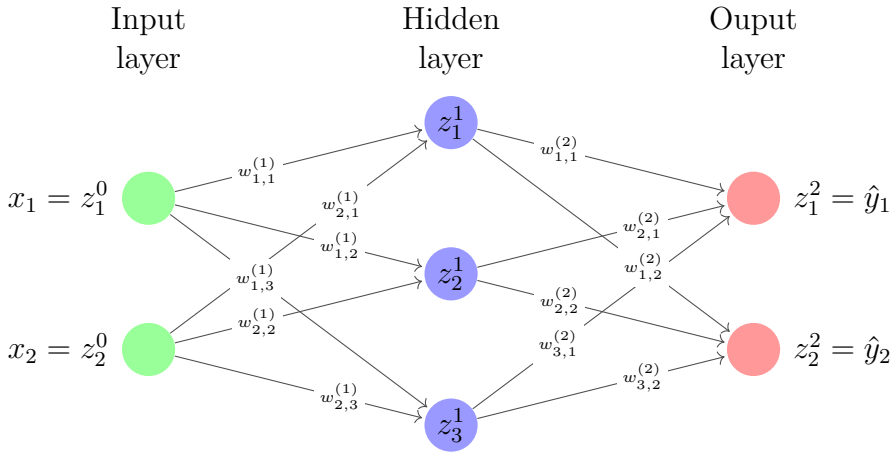


Figure 2.2: A feedforward neural network (FNN) with the weights $\boldsymbol{w}$. This FNN takes an input $\boldsymbol{x} \in \mathbb{R}^2$ and gives an output $\hat{\boldsymbol{y}} \in \mathbb{R}^2$.

A feedforward neural network (FNN) is a type of neural networks where signals only are transmitted from one layer to the subsequent layer. A FNN is shown in Figure 2.2. For illustrative purposes the layers are usually arranged from the left (input) to the right (output). The first layer contains no neurons but holds all the input variables, which are transmitted as inputs to the neurons of the next layer, and it is called the input layer. The final layer, which is called the output layer, contains neurons with an observable signal. The layers between the input and output layers are called hidden layers. These layers contain arbitrary many neurons. The input from the input layer propagates through the network from left to right, which then yields the observable output from the output layer. Furthermore, the connections between neurons are one-way directed, and information is transmitted from neurons in the current layer to neurons in the subsequent layer (to the right). This means that in a FNN there are no connections within a layer, that no connections skip a layer and finally, that no connections go backwards.

This architecture means that each neuron receives inputs from all neurons of the previous layer, and outputs a value to the neurons in the next layer. Note that multiple output arrows from a neuron in a network diagram, such as seen in Figure 2.2, merely illustrate that the output is transmitted to several neurons. The output of a neuron j in the first hidden layer is determined by weighting the different inputs as

$$z_j^{(1)} = f\left(\sum_{i=0}^{n} w_{i,j}^{(1)} z_i^{(0)}\right),$$

where $z_j^{(1)}$ is the output of neuron j in the first hidden layer, n is the number of inputs, $z_i^{(0)} = x_i$ is the value corresponding to input $i \in \{0, \ldots, n\}$, f is called the activation function, Section 2.4.4, and $w_{i,j}^{(1)}$ is the weight corresponding to the connection between input i and neuron j in the first hidden layer. Usually, $w_{0,j}^{(l)}$, for an arbitrary layer (l), is called the bias and it does not obtain information from any other neuron or external data, which means that $x_0 = 1$.

By changing the weights, the network is able to obtain different outputs for a specific external input data. This enables the possibility to train a neural network to yield a specific output for a specific input data to the neural network. In fact, the universal approximation theorem states that all continuous functions can be approximated by a feedforward network with a single hidden layer, which contains a sufficient number of neurons, with an arbitrary activation function. Among others, this has been shown by [35] and [36] for mild assumptions on the activation function.

2.4.2 Stochastic Gradient Descent

When training a neural network, we want the neural network to produce a specific output $\hat{\boldsymbol{y}}$ for a specific input $\boldsymbol{x}$ fed to the network [25], [37]. Therefore, the objective is to find the network weights that minimizes a loss function $L(\boldsymbol{y}, \hat{\boldsymbol{y}})$ between the output $\hat{\boldsymbol{y}}$ of the network and the desired output $\boldsymbol{y}$, e.g., the squared error SE $= ||\hat{\boldsymbol{y}} - \boldsymbol{y}||_2^2$. Note that usually the data consists of several input and output pairs that the neural network should learn. Hence the loss function to be minimized is given by

$$\mathcal{L} = \sum_{n=1}^{N} L(\boldsymbol{y}^{(n)}, \hat{\boldsymbol{y}}^{(n)}),$$

where the superscript (n) denotes the n-th input pattern. This minimization problem is usually impossible to solve analytically. Therefore, a common approach to minimize the loss is to use gradient descent and iteratively update the weights according to

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla \mathcal{L}\Big(\boldsymbol{w}^{(\tau)}\Big),$$

where $\eta > 0$ is known as the learning rate and $\boldsymbol{w}^{(\tau)}$ is weights for the τ-th iteration. This approach changes the weights in the direction of the greatest decrease of the loss function. This method finds a local minimum of the loss and, therefore, it is not guaranteed that this method converges to a global minimum. Hence, in order to find a sufficiently good minimum, it can be necessary to run the gradient descent several times for different initial weights, which is impractical [25].

A practically useful and popular method, which can be seen as a stochastic approximation of gradient descent, is stochastic gradient descent (SGD) (see Algorithm 3). The randomness is introduced by not taking the derivative of $\mathcal{L}$, but rather of the loss from a randomly picked input pattern $L(\boldsymbol{y}^{(n)}, \hat{\boldsymbol{y}}^{(n)})$ and update the weights based on this derivative. This does not necessarily result in a decrease in total loss, which thereby reduces the risk of getting stuck at a local minimum. However, this might introduce too much noise into the gradient descent, resulting in a network that fails to converge, so usually the SGD direction is an average over the directions

of a small random subset of data points [37]. These random subsets are commonly known as *mini-batches*.

Algorithm 3 Stochastic gradient descent

1: Choose initial weights $\boldsymbol{w}$ and learning rate η
2: **for** $\tau = 1, \ldots, T$ **do**
3: Shuffle the data in the training set with N pairs of inputs and outputs
4: **for** $n = 1, \ldots, N$ **do**
5: Update the weights by $\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla L(\boldsymbol{y}^{(n)}, \hat{\boldsymbol{y}}^{(n)})$
6: **end for**
7: **end for**

2.4.3 Backpropagation

In order to use stochastic gradient descent, an efficient method for evaluation of the gradients of the loss functions is necessary. Backpropagation provides such technique for evaluating the gradients of the loss for each input and output pair, i.e., $\nabla L(\boldsymbol{w})$ where $\boldsymbol{w}$ is the weights in the network, by propagating information forwards and backwards through the network [25], [37]. In particular, it provides an approach for evaluating each partial derivative of the gradient.

Suppose a simple linear model for an output $\hat{y}_k$, for an arbitrary input/output pair, as a linear combination of input $\boldsymbol{x} = x_1, \ldots, x_t$ with weights $\boldsymbol{w}_1 = w_{k,1}^{(1)}, \ldots, w_{k,t}^{(1)}$ for layer (1) (note that this simple linear model only consists of one layer between the input and output) such that

$$\hat{y}_k = \sum_i w_{k,i}^{(1)} x_i.$$

Also, suppose that we have a sum-of-squares loss, given by

$$L = \frac{1}{2} \sum_k (\hat{y}_k - y_k)^2,$$

where y_k is the desired output. Then the partial derivative of $\mathcal{L}$ with respect to $w_{k,i}^{(1)}$ is given by

$$\frac{\partial L}{\partial w_{k,i}^{(1)}} = \frac{\partial L}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial w_{k,i}^{(1)}} = (\hat{y}_k - y_k) x_i, \tag{2.15}$$

since the relationship between L and $w_{k,i}$ is through the output $\hat{y}_k$.

Compared to a neural network, the simple example above lacks both hidden layers and activations. For a neural network, the input to neuron k is instead an activation of a neuron from the previous layer, i.e., the output from a neuron of the previous layer sent through an activation function f. Hence, if $z_i^{(l)}$ is the activation from neuron i in the previous layer (l), the input to neuron k in the current layer $(l+1)$ is given by

$$a_k^{(l+1)} = \sum_i w_{i,k}^{(l+1)} z_i^{(l)}, \tag{2.16}$$

where $z_k^{(l+1)} = f\left(a_k^{(l+1)}\right)$. Utilizing the chain rule for partial derivatives, and the fact that (L) depends on the weight $w_{j,k}^{(l+1)}$ only via the summed input $a_k^{(l+1)}$, gives that

$$\frac{\partial L(\boldsymbol{w})}{\partial w_{j,k}^{(l+1)}} = \frac{\partial L(\boldsymbol{w})}{\partial a_k^{(l+1)}} \frac{\partial a_k^{(l+1)}}{\partial w_{j,k}^{(l+1)}}. \tag{2.17}$$

Evaluating the second term on the right hand side in Equation (2.17) yields

$$\frac{\partial a_k^{(l+1)}}{\partial w_{j,k}^{(l+1)}} = \frac{\partial}{\partial w_{j,k}^{(l+1)}} \sum_i w_{i,k}^{(l+1)} z_i^{(l)} = z_j^{(l)}. \tag{2.18}$$

Thus, the first factor in the evaluation of the partial derivatives for layer $(l+1)$ is known if activation $z_i^{(l)}$ of the previous layer is known.

For the output layer (L) the first term on the right hand side in Equation (2.17) evaluates to

$$\frac{\partial L(\boldsymbol{w})}{\partial a_i^{(L)}} = \hat{y}_i - y_i,$$

where $\hat{y}_i$ and y_i is the output of the network and the desired output, respectively. This follows from the obtained result in Equation (2.15). By once again utilizing the chain rule for partial derivatives, the corresponding partial derivatives for the next layer $(L-1)$ can then be evaluated by

$$\frac{\partial L(\boldsymbol{w})}{\partial a_k^{(L-1)}} = \sum_i \frac{\partial L(\boldsymbol{w})}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial a_k^{(L-1)}},$$

where i is summed over all connections from layer $(L-1)$ to neuron k in layer (L). By continuing to utilize the chain rule and using the same state of mind, the partial derivative of the loss with the respect to an activation of an arbitrary layer (l) can be evaluated by the backpropagation formula

$$\begin{aligned} \frac{\partial L(\boldsymbol{w})}{\partial a_k^l} &= \sum_i \frac{\partial L}{\partial a_i^{(l+1)}} \frac{\partial a_i^{(l+1)}}{a_k^{(l)}} = \sum_i \frac{\partial L}{\partial a_i^{(l+1)}} \frac{\partial \sum_j w_{j,i}^{(l+1)} f\left(a_j^{(l)}\right)}{\partial a_k^{(l)}} \\ &= f'\left(a_k^{(l)}\right) \sum_i w_{k,i}^{(l+1)} \frac{\partial L(\boldsymbol{w})}{\partial a_i^{(l+1)}}, \end{aligned} \tag{2.19}$$

where f' is the derivative of the activation function f, which needs to be at least once differentiable at the inputs.

Thus, by iteratively calculating the partial derivatives of the losses w.r.t. the activations from the last to the first layer, the backpropagation formula in Equation (2.19) provides a technique for evaluating the gradient of the loss w.r.t. the weights $\boldsymbol{w}$. The partial derivatives of this gradient are obtained by using Equation (2.18) and (2.19) in Equation (2.17). Doing so yields

$$\frac{\partial L(\boldsymbol{w})}{\partial w_{j,k}^{(l)}} = \frac{\partial L(\boldsymbol{w})}{\partial a_k^{(l)}} \frac{\partial a_k^{(l)}}{\partial w_{j,k}^{(l)}} = z_j^{(l-1)} f'\left(a_j^{(l)}\right) \sum_i w_{j,i}^{(l+1)} \frac{\partial L(\boldsymbol{w})}{\partial a_i^{(l+1)}}.$$

This result implies that by propagating the input forward through the network and then backwards to calculate the partial derivatives of the loss, backpropagation can be used to calculate the gradient of the loss. These gradients are then used to update the network to minimize the total loss of the network. What makes backpropagataion so efficient is that the updates are local, i.e., they only depend on the weights in the current layer and the state of the neuron that feeds input to the layer.

When utilizing mini-batches, the derivative of the total loss can be evaluated by repeating the above steps for each pair of input and output in the batch, and then summing over all input/output pairs m in the batch. This means that

$$\frac{\partial \mathcal{L}}{\partial w_{j,i}^{(l)}} = \sum_m \frac{\partial L(\boldsymbol{y}^{(m)}, \hat{\boldsymbol{y}}^{(m)})}{\partial w_{j,i}^{(l)}}$$

if it is assumed that each hidden or output neuron has the same activation function f; note that it can be generalized to individual activation functions for each neuron by keeping track of which activation function goes with which neuron [25].

2.4.4 Activation

For the simplest model of a biological neuron, the artificial neuron only has two states: active and inactive [37]. An active and inactive neuron would then output $y = 1$ and $y = 0$, respectively. A reasonable choice of activation function f would then be the Heaviside step function H

$$H(x - \mu) = \begin{cases} 0 & \text{if } x - \mu < 0 \\ 1 & \text{if } x - \mu \geq 1 \end{cases},$$

where μ is a threshold that the neuron has to reach to be active. However, to train the neuron to give a specific output for a specific input using backpropagation is not possible in this case since the derivative of the Heaviside step function is zero at all points except for the point zero, where its derivative is undefined. Therefore, we want an activation function with a finite derivative; while it is also preferable to let neurons have a continuous output, since then it is possible to model neurons that output varying "strength".

The sigmoid function

$$S(x) = \frac{1}{1 + e^{-x}}$$

is a function with promising features since it has a finite derivative and are approximately equal to 1 and 0 for large positive and negative input values, respectively. However, utilizing backpropagation with sigmoid as activation function can give rise to the vanishing-gradient problem [37]. This is a problem where partial derivatives $\frac{\partial L(\boldsymbol{w})}{\partial w_{j,k}^{(l)}}$ obtain values close to zero. The problem is usually worse for layers early in neural networks with multiple hidden layers, since the products of the derivatives of the activation function $f'\left(a_k^{(l)}\right)$ in Equation (2.19) vanishes quickly as (l) decreases.

The problem arises due to the fact that the maximum derivative of the sigmoid function is $\frac{1}{2}$. For a network with many layers these derivatives will be multiplied, hence the partial derivatives will become increasingly small as (l) decreases. A consequence of this is that the early layers will learn very slowly or even fail to learn at all.

To avoid the aforementioned problem, [38] suggested the usage of rectified linear units (ReLUs), which are neurons with the following ReLU activation function

$$\mathrm{ReLU}(x) = \max(0, x).$$

This provides a non-decreasing function which is differentiable everywhere except at zero, where its derivative is undefined. For practical reasons, the derivative at zero is often set to zero. Also, $\mathrm{ReLU}'(x) = 1$ for $x > 0$. This fixes the vanishing-gradient problem but gives rise to the dying ReLU problem. The dying ReLU problem arises from the fact that ReLUs can be pushed to states where they are inactive for all inputs since ReLU outputs zero for all non-positive inputs. That all neurons are inactive is obviously not desirable. However, an advantage of ReLU is that it can yield a sparse network, i.e., a network where many of the hidden neurons are inactive, which is believed to have several desirable properties [37]. Therefore, this is one of the most common activation function today.

A common activation function for the neurons of the output layer is the softmax function. Outputs from the softmax activation function is defined by

$$\sigma(a_1, \ldots, a_K)_i = \frac{e^{ca_i}}{\sum_{j=1}^{K} e^{ca_j}}, \quad i \in \{1, \ldots, K\},$$

where c is an arbitrary constant and a_i is the input to neuron i of the output layer. The normalization indicates that the output can be interpreted as a probability distribution with K different possible outcomes.

2.4.5 Dropout

A common problem when training a neural network to learn a specific task, is that the network is prone to overfitting. A technique that has shown to reduce overfitting is dropout [37], [39]. This technique handles overfitting by randomly deactivating neurons, with a predetermined probability $(1 - p)$, during training. By temporarily deactivating these neurons they will produce no output, regardless of what the input to these neurons were. The idea of dropout is that the learnt patterns have to be robust under random deactivation of hidden neurons, thereby reducing overfitting.

That a neuron is deactivated means that the corresponding row in the weight matrix is set to zero. Let $\boldsymbol{m}^{(i)}$ denote the weight matrix of layer i, before dropout. The corresponding weight matrix $\boldsymbol{w}^{(i)}$, which is obtained after dropout, is then given by

$$\boldsymbol{w}^{(i)^T} = \boldsymbol{m}^{(i)^T} \cdot \mathrm{diag}([\epsilon_{i,j}]_{j=1}^{K_i}), \quad \epsilon_{i,j} \sim \mathrm{Bernoulli}(p_i),$$

where K_i denotes the size of the layer and $p_i \in [0, 1]$ is the probability that the neuron will be retained. $\boldsymbol{w}^{(i)}$ will be used during training, and during validation and testing the scaled full matrix $p_i \boldsymbol{m}^{(i)}$ will be used [39].

2.4.6 Bayesian Neural Networks

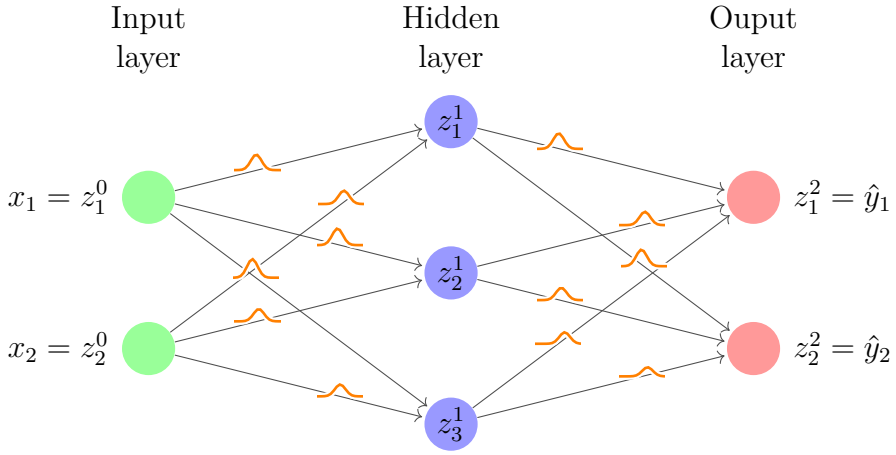


Figure 2.3: A Bayesian neural network (BNN) where the weights are represented by probability distributions. This BNN takes an input $\boldsymbol{x} \in \mathbb{R}^2$ and gives a predicted output $\hat{\boldsymbol{y}} \in \mathbb{R}^2$.

Bayesian neural networks (BNNs) have an architecture that is mostly similar to traditional neural networks. However, a fundamental difference between BNNs and frequentist networks (Section 2.4.1) is that the weights $\boldsymbol{w}$ are represented by probability distributions $\pi(\boldsymbol{w})$, as illustrated in Figure 2.3, instead of point estimates [26], [30]. Therefore in BNNs one tries to learn the posterior of the weights $\pi(\boldsymbol{w}|\mathcal{D})$, instead of the point estimated weights that best explains the data $\mathcal{D}$. Some of the benefits of BNNs are that they are less likely to overfit, since the learnt pattern must be robust under variations in the weights $\boldsymbol{w}$, and that they allow for uncertainty estimation of the predictions [26], [30].

BNNs are based on Bayes' theorem, Equation (2.1), where $\pi(\boldsymbol{w})$ represents prior belief about the network's weights. The probability distribution of interest is the posterior of the weights $\pi(\boldsymbol{w}|\mathcal{D})$. If the posterior is known then one can compute the posterior predictive distribution, Equation (2.3), for a new input $\boldsymbol{x}^*$. To use the posterior predictive is equivalent to compute the prediction of an infinite number of neural networks since the prediction is based on all possible values of $\boldsymbol{w}$, weighted by the posterior probability $\pi(\boldsymbol{w}|\mathcal{D})$ of that value. Unfortunately, for neural networks the space of potential weights $\mathcal{W}$ is usually vast and the posterior is complex, making the integral in Equation (2.3) intractable [30]. Therefore, one needs to approximate this integral, which can be achieved by MC integration

$$\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx \frac{1}{T}\sum_{t=1}^{T} \pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}^{(t)}), \quad \boldsymbol{w}^{(t)} \sim \pi(\boldsymbol{w}|\mathcal{D}). \tag{2.20}$$

This means that the posterior predictive can be approximated by T forward passes through the network, where each forward pass samples new weights from the posterior distribution.

For most Bayesian neural networks, the posterior $\pi(\boldsymbol{w}|\mathcal{D})$ does not have an analytic representation, and furthermore sampling from this distribution can be difficult

with traditional MCMC methods [40]. Therefore, a variational Bayesian approach is often used, where the posterior is approximated by a tractable posterior $q(\boldsymbol{w}|\theta)$ with distribution parameters θ, as described in Section 2.1.4. Note that this means that the network tries to learn θ. A consequence of this is that the number of learnable parameters in the network usually increases since many distributions are parameterized by more than one parameter.

2.4.6.1 Bayes by Backprop

In *Bayes by Backprop* the network is parameterized by the variational posterior parameters θ rather than by weights, which is the norm [30]. The goal is to find the parameters θ^{opt} which make the variational distribution as similar as possible to the true posterior, see Equation (2.6).

One essential question which is yet to be answered is how the network should learn the parameters of the variational posterior. The optimization problem in Equation (2.6) is generally not possible to solve exactly. Therefore, Bayes by Backprop uses gradient descent to find an approximation of the optimum by the use of backpropagation [30]. Thus, it is necessary to compute

$$\nabla_\theta \mathcal{F}(\mathcal{D}, \theta) = \nabla_\theta \mathbb{E}_{q(\boldsymbol{w}|\theta)} \left[\log \left(\frac{q(\boldsymbol{w}|\theta)}{\pi(\boldsymbol{w})} \right) - \log \left(\pi(\mathcal{D}|\boldsymbol{w}) \right) \right]. \tag{2.21}$$

However, this poses a problem since the gradient is taken with respect to the parameters of the distribution over which the expectation is taken.

This problem can be avoided by utilizing a different sampling method for $\boldsymbol{w}$. The idea is to reparameterize the random weights $\boldsymbol{w} \sim q(\boldsymbol{w}|\theta)$ as deterministic weights $\boldsymbol{w} = t(\theta, \epsilon)$. Where $t(\theta, \epsilon)$ is a deterministic function that transforms the parameters θ and some auxiliary noise $\epsilon \sim p(\epsilon)$, whose marginal is independent of θ, to a sample from $q(\boldsymbol{w}|\theta)$. Furthermore, in order for the reparameterization to be useful $t(\theta, \epsilon)$ also needs to be differentiable w.r.t. θ. This transformation is commonly known as a *Reparameterization Trick* [41]. The reparameterization allows for Equation (2.21) to be expressed as [30]

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{q(\boldsymbol{w}|\theta)} \left[\log \left(\frac{q(\boldsymbol{w}|\theta)}{\pi(\boldsymbol{w})} \right) - \log \left(\pi(\mathcal{D}|\boldsymbol{w}) \right) \right] = \\
&= \nabla_\theta \mathbb{E}_{p(\epsilon)} \left[\log \left(\frac{q(\boldsymbol{w}|\theta)}{\pi(\boldsymbol{w})} \right) - \log \left(\pi(\mathcal{D}|\boldsymbol{w}) \right) \right] \\
&= \mathbb{E}_{p(\epsilon)} \left[\nabla_\theta \left(\log \left(\frac{q(\boldsymbol{w}|\theta)}{\pi(\boldsymbol{w})} \right) - \log \left(\pi(\mathcal{D}|\boldsymbol{w}) \right) \right) \right].
\end{aligned}$$

The approximated loss, Equation (2.8), can then also be expressed as

$$\mathcal{F}(\mathcal{D}, \theta) \approx \frac{1}{T} \sum_{t=1}^{T} \log \left(\frac{q(\boldsymbol{w}^{(t)}|\theta)}{\pi(\boldsymbol{w}^{(t)})} \right) - \log \left(\pi(\mathcal{D}|\boldsymbol{w}^{(t)}) \right), \tag{2.22}$$

where $\boldsymbol{w}^{(t)} = t(\theta, \epsilon)$ and $\epsilon \sim p(\epsilon)$. For mini-batch optimization with M batches [40]

suggested the loss

$$\begin{aligned}\mathcal{F}_i^{\text{batch}}(\mathcal{D}_i, \theta) &= \frac{1}{M}\mathbb{E}_{q(\boldsymbol{w}|\theta)}\left[\log\left(\frac{q(\boldsymbol{w}|\theta)}{\pi(\boldsymbol{w})}\right)\right] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}\left[\log\left(\pi(\mathcal{D}_i|\boldsymbol{w})\right)\right] \\ &\approx \frac{1}{M \cdot T}\sum_{t=1}^{T}\log\left(\frac{q(\boldsymbol{w}^{(t)}|\theta)}{\pi(\boldsymbol{w}^{(t)})}\right) - \frac{1}{T}\sum_{t=1}^{T}\log\left(\pi(\mathcal{D}_i|\boldsymbol{w}^{(t)})\right),\end{aligned}$$

for batch i, where again $\boldsymbol{w}^{(t)} = t(\theta, \epsilon)$ and $\epsilon \sim p(\epsilon)$. Note that $\mathcal{D}_i$ denotes a random batch of data and that $\sum_{i=1}^{M} \mathcal{F}_i^{\text{batch}}(\mathcal{D}_i, \theta) = \mathcal{F}(\mathcal{D}, \theta)$.

What this reparameterization trick effectively does is that it translates the stochasticity of the model from the sought parameters θ to an auxiliary noise variable ϵ, thereby essentially making the loss dependence on θ deterministic. Furthermore, the values of θ are now incorporated into each weight, due to $t(\theta, \epsilon)$. The usefulness of this result is that the approximation of the expectation, Equation (2.22), now is differentiable w.r.t. θ, since the expectation is estimated by samples which are generated by an auxiliary variable that is independent of θ [41], [30]. This allows for the parameters to be learnt by backpropagation, by computing the gradient of the loss in Equation (2.22).

If one assumes that the variational posterior is a distribution parameterized by location and scale, e.g., Gaussian or Laplace, then the reparameterization $t(\theta, \epsilon)$ takes the form $t(\theta, \epsilon) = \text{location} + \text{scale} \odot \epsilon$ where ϵ are generated from the standard of that distribution and $\odot$ denotes element-wise multiplication. For a diagonal Gaussian distribution $\theta = (\mu, \sigma)$ this means that the weights are given by $\boldsymbol{w} = t(\theta, \epsilon) = \mu + \sigma \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. The fact that the variational distribution is a diagonal Gaussian also means that the number of parameters in the network is doubled compared to the FNNs described in Section 2.4.1. This is because each connection now is parameterized by a mean μ and a variance σ^2, instead of just a value of the weight.

2.4.6.2 Monte Carlo Dropout

Monte Carlo dropout (*MC dropout*) is a variational approach, which means that the true posterior is approximated with the variational distribution $q(\boldsymbol{w}|\theta)$. The distribution is based on dropout, where nodes randomly are equated with zero with probability $(1 - p)$. This means that the distribution randomly equates rows of the networks weight matrices with zero. $q(\boldsymbol{w}|\theta)$ is independent between the layers, so for layer i the weights $\boldsymbol{w}^{(i)}$ sampled from $q(\boldsymbol{w}|\theta)$ are given by

$$\boldsymbol{w}^{(i)^T} = \boldsymbol{m}^{(i)^T} \cdot \text{diag}([\epsilon_{i,j}]_{j=1}^{K_i}), \quad \epsilon_{i,j} \sim \text{Bernoulli}(p_i), \tag{2.23}$$

where K_i denotes the size of the layer, $\boldsymbol{w}^{(i)}$ are the random weights of layer i and $p_i \in [0, 1]$ is the probability that the node will not be dropped. The sought parameters for $q(\boldsymbol{w}|\theta)$ are simply the weight matrix $\boldsymbol{m}^{(i)}$ of each layer, i.e., $\theta = (\boldsymbol{m}^{(1)}, \ldots, \boldsymbol{m}^{(L)})$.

The MC dropout method assumes that the prior on the weights is $\pi(\boldsymbol{w}) = \mathcal{N}(0, l^{-2}I)$, where l is a scaling parameter. This choice of prior and variational posterior implies that the complexity loss in Equation (2.7) does not exist [42]. However, this can be

avoided by utilizing that each factor of $q(\boldsymbol{w}|\theta)$ can be approximated by a mixture of two Gaussians with small variances and one mean fixed to zero. This means that the distribution for weight vector j in layer i can be approximated as

$$\boldsymbol{w}_{j,\cdot}^{(i)} = p_i \mathcal{N}(\boldsymbol{m}_{j,\cdot}^{(i)}, \sigma^2 I) + (1 - p_i)\mathcal{N}(0, \sigma^2 I),$$

where $\boldsymbol{m}_{j,\cdot}^{(i)}$ denotes the weight vector that describes all connections from node j in layer i. The complexity loss in Equation (2.7) can then be approximated as [42]

$$\mathrm{KL}\left(q(\boldsymbol{w}|\theta)\|\pi(\boldsymbol{w})\right) \approx \sum_{i=1}^{L} \frac{p_i l^2}{2} \|\boldsymbol{m}^{(i)}\|_2^2 + \frac{l^2}{2}\|\boldsymbol{b}^{(i)}\|_2^2,$$

where N is the number of data points, L is the number of layers and $\boldsymbol{b}_i$ are the biases of layer i. A detailed derivation of this result can be found in [42]. Furthermore, approximating the likelihood loss in Equation (2.7) with a sample $\boldsymbol{w} \sim q(\boldsymbol{w}|\theta)$ yields

$$\begin{aligned}\mathbb{E}_{q(\boldsymbol{w}|\theta)}\left[\log(\pi(\mathcal{D}|\boldsymbol{w}))\right] &= \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{w}|\theta)}\left[\log(\pi(y_n|x_n, \boldsymbol{w}))\right] \\ &\approx \sum_{n=1}^{N} \log(\pi(y_n|x_n, \boldsymbol{w}_n)).\end{aligned}$$

These two approximations means that the loss in Equation (2.7) takes the form

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{L}\left(\frac{p_i l^2}{2}\|\boldsymbol{m}^{(i)}\|_2^2 + \frac{l^2}{2}\|\boldsymbol{b}^{(i)}\|_2^2\right) - \sum_{n=1}^{N}\log(\pi(y_n|x_n, \boldsymbol{w}_n)).$$

Scaling the result with $\frac{1}{N}$ does not change the solution, therefore,

$$\begin{aligned}\mathcal{F}(\mathcal{D}, \theta) &\propto \frac{1}{N}\sum_{i=1}^{L}\left(\frac{p_i l^2}{2}\|\boldsymbol{m}^{(i)}\|_2^2 + \frac{l^2}{2}\|\boldsymbol{b}^{(i)}\|_2^2\right) - \frac{1}{N}\sum_{n=1}^{N}\log(\pi(y_n|x_n, \boldsymbol{w}_n)) \\ &= -\frac{1}{N}\sum_{n=1}^{N}\log(\pi(y_n|x_n, \boldsymbol{w}_n)) + \sum_{i=1}^{L}\frac{p_i l^2}{2N}\left(\|\boldsymbol{m}^{(i)}\|_2^2 + \frac{1}{p_i}\|\boldsymbol{b}^{(i)}\|_2^2\right).\end{aligned} \tag{2.24}$$

The first term is the negative log likelihood of the model output. The second term is equivalent to weight decay with decay constant $\frac{p_i l^2}{2N}$. Lastly, note that since the stochasticity of the network is in the auxiliary random variable $\epsilon \sim \text{Bernoulli}(p)$ and not in the sought parameters, one can easily take the derivative of the loss in Equation (2.24) w.r.t. the variational parameters. This follows from the definition of $q(\boldsymbol{w}|\theta)$, so that the weights are given by a deterministic transformation of the deterministic weights and the auxiliary ϵ, Equation (2.23). This means that the parameters, i.e., the weights of the model, are easily learnt through backpropagation.

This derivation illustrates that simply applying dropout to the model yields an approximate Bayesian model. Furthermore, if the model uses weight decay then this corresponds to a prior belief on the weights. That the dropout can be interpreted as a Bayesian model also means that the model can be used for variational inference, where Equation (2.20), is aproximated by sampling from the variational distribution $q(\boldsymbol{w}|\theta)$ instead of the true posterior $\pi(\boldsymbol{w}|\mathcal{D})$. The difference between MC dropout and regular dropout, Section 2.4.5, is that the former drops nodes even during testing [42], thereby allowing variational inference.

2.5 Matrix Factorization

This section introduces probabilistic Matrix factorization. This method will be used in active learning, where the model suggests which points to add to the data set. Matrix factorization predicts the data by finding relationships between the rows and columns (or corresponding entities in the higher-dimensional cases) in a matrix (or tensor in the higher-dimensional cases). Each dimension in the matrix represents different reaction components and the entries are the associated yields.

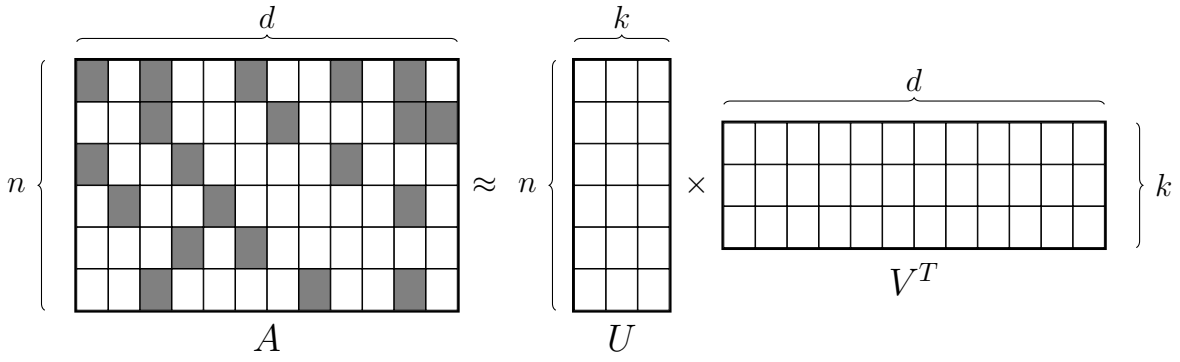


Figure 2.4: The idea of matrix factorization (MF), where $A \in \mathbb{R}^{n \times d}$ is approximated by the decomposition of $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{d \times k}$. The filled entries are known.

Matrix factorization (MF) obtained its high popularity due to the Netflix Prize competition, where the aim was to create a model for recommending movies that a user may like based on the rating of previously watched movies [43]. This competition established the MF method as a very potent model for recommender systems, which aim at providing personalized recommendations based on previous actions [43].

The recent MF models are based on the same idea as was proposed already in 2003 by [44]. They proposed a MF model for partially observed matrices, which enabled the possibility to use MF for predictive machine learning [45]. The objective of [44] was to minimize the weighted Frobenius distance

$$J(X) = \sum_{i,a} W_{i,a}(X_{i,a} - A_{i,a})^2$$

to obtain a low-rank approximation of the target matrix $A \in \mathbb{R}^{n \times d}$, where each dimension represents a feature in the data (e.g., two categorical variables representing different solvents and ligands). Note that $W \in \mathbb{R}_+^{n \times d}$ is a non-negative weight matrix and $X \in \mathbb{R}^{n \times d}$ is a matrix of rank $k \leq \max(n, d)$. From a MF perspective, the decomposition $X = UV^T$, where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{d \times k}$, is considered when approximating A. The objective is then to find the matrices U and V that minimize the Frobenius distance. Using weights enables the model to consider a target matrix with unobserved entries, where a zero and one weight corresponds to a unobserved and observed entry, respectively. This made it possible to predict

unobserved entries. The general idea of matrix factorization is illustrated in Figure 2.4.

For the unweighted case (i.e., W is a matrix of ones), the Frobenius distance has a zero gradient with respect to U and V if and only if the columns of U and V are spanned by eigenvectors of AA^T and A^TA, respectively; where the global minimum can be obtained when U and V are spanned by the eigenvectors corresponding to the highest eigenvalues [44]. For the weighted case, the problem to minimize the Frobenius distance lacks an analytic solution. Therefore, [44] suggests and describes an Expectation-Maximization procedure, which they state to be more simple and cost effective compared to other previously described procedures. In the expectation step, values from the current estimation of X are used to fill the missing values of A to create A' (containing no missing values); and in the maximization step, X is estimated as a low-rank approximation (LRA) of A'

$$X^{(t+1)} = \mathrm{LRA}_k\Big(W \otimes A + (\mathbf{1} - W) \otimes X^t\Big),$$

where $\mathrm{LRA}_k(Z)$ is the unweighted rank-k approximation of the matrix Z and the weight matrix W consists of weights between zero and one. In general, an unweighted rank-k approximation of the matrix Z is obtained by solving the following problem

$$\begin{aligned} \min_{Z'} \quad & ||Z - Z'||, \\ \text{s.t.} \quad & \mathrm{rank}(Z') \leq k, \end{aligned}$$

where $||\cdot||$ is an arbitrary norm and $k \leq \mathrm{rank}(Z)$. When minimizing the Frobenius norm, the solution is computed by utilizing the singular value decomposition (SVD) of Z [46].

Even though [44] found their method effective in many cases, in some cases they found it to converge to a local minimum which is not a global minimum. Since, for a deterministic method, initialization plays a crucial role for the convergence to a global minimum, they also suggested several procedures to initialize X. One of these procedures, which they found very effective, was to initialize X to zero and in the first step find a full rank approximation; and thereafter decrease the rank of the approximations.

2.5.1 Probabilistic Matrix Factorization

In 2008, [47] improved the work of [44] by introducing a probabilistic approach. This approach assumed a conditional multivariate Gaussian distribution over the observed entries in the target matrix, a zero-mean spherical Gaussian prior on the decomposition matrices U and V and also a posterior distribution over the decomposition matrices. When maximizing the log-posterior, they obtained a regularized minimization problem with the objective to minimize the Frobenius distance

$$\min_{U,V} \sum_{i,j} I_{ij} \left(A_{i,j} - \sum_k U_{i,k} V_{j,k} \right)^2 + \lambda_V ||V||_F^2 + \lambda_U ||U||_F^2,$$

where k is the desired rank of the low-rank approximation of A, $||\cdot||_F$ is the Frobenius norm, $\lambda_U, \lambda_V > 0$ are regularization coefficients and $I_{i,j}$ is the indicator function which is equal to one if element (i, j) is observed in X and zero if it is unobserved. Moreover, using steepest descent, they were able to perform probabilistic matrix factorization (PMF) in linear time with respect to the number of observations to find a local minimum to the minimization problem. Also, they gave suggestions on how to choose the regularization parameters and handle features with very few entries. Furthermore, they discussed that a fully Bayesian approach would be more computationally expensive but that preliminary results suggest that such an approach would result in a significant improvement in predictive accuracy.

Therefore, [48] extend their model by presenting a Bayesian probabilistic matrix factorization model using a MCMC method. They placed Gaussian priors on the matrices U and V, and placed Gaussian-Wishart priors on the hyperparameters. In their work, the predictive distribution of estimated target matrix A^* is approximated using MC integration

$$\pi(A^*|A, \theta_0) \approx \frac{1}{K} \sum_{k=1}^{K} \pi(A^*|U^{(k)}, V^{(k)}),$$

where θ_0 is the hyperparameters of the parameters describing U and V; and $(U^{(k)}, V^{(k)})$ are sampled using Gibbs sampling. For more details about the assumed distributions, the Gibbs sampler and so forth, see [48].

2.5.2 Macau

Macau, which is applied in this thesis, is a method for factorizing heterogeneous data which was proposed by [45]. They continued the work of [48] by using the same idea of a fully Bayesian approach but also added some additional features. In particular, Macau has the ability to handle multiple relations between factors (e.g., ligands, solvents, bases and additives). In order to handle these multiple relations between factors, Macau considers a relational model with a set of features $\mathcal{E}$ and a set of relations $\mathcal{R}$ to enable each relation $R \in \mathcal{R}$ to link together two or more features. This enables Macau to perform tensor factorization, where a tensor of arbitrary rank (also known as order or degree) is predicted, instead of only matrix factorization. The rank of a tensor is the number of indices (dimensions) that is need for the tensor to identify all factors. That is, the rank of the tensor is the number of factors, i.e., the number of different reaction components in the data. Moreover, Macau has the ability to incorporate side information (features of instances of a factor) for any factor and relations between them. In particular, by using side information one can add extra information about the instances of the factors. For instance, this can be information of the molecular substructure in the form of a sparse binary vector, i.e., a chemical fingerprint, see Section 2.7.1. Macau uses this extra information for the predictions of latent vectors in order to obtain more accurate factorization. As for the work by [48], to sample from the posterior of the model parameters Macau utilizes Gibbs sampling. This approach was implemented using C++ and published online.[1]

[1] `https://github.com/jaak-s/macau`

2.6 Kennard-Stone Algorithm

The Kennard-Stone algorithm [49] is a sequential algorithm that aims to pick a new data point that maximizes the minimum distance to all selected points. This algorithm will be used as an alternative to random sampling, where new points are selected randomly, in order to be compared to uncertainty sampling, see Section 3.1.3. In each iteration i a new point is selected by solving the following problem

$$\arg\max_{v \in S \setminus S_{i-1}} \min_{j \in S_{i-1} \setminus \{v\}} d_{v,j},$$

where $S := \{v_1, \ldots, v_n\}$ is the set of all points that can be selected, S_{i-1} is the selected points in iteration $i-1$ and $d_{v,j}$ is the distance between point v and j. In each step, the algorithm determines the distances between the selected points and the unselected points, and then selects the unselected point with the greatest minimum distance to the selected points. As initial points, the two points that are farthest apart are selected. Thus, solving

$$\arg\max_{v,j;\ v \neq j} d_{v,j}.$$

Note that there is no guarantee of uniqueness of S_i since there may be more than one candidate point whose minimum distance to a selected point yields the same maximum. To break this tie, [49] suggested choosing point v with the smallest index among the points with the same minimum distance to already selected points.

2.7 Representation of Molecules and Reactions

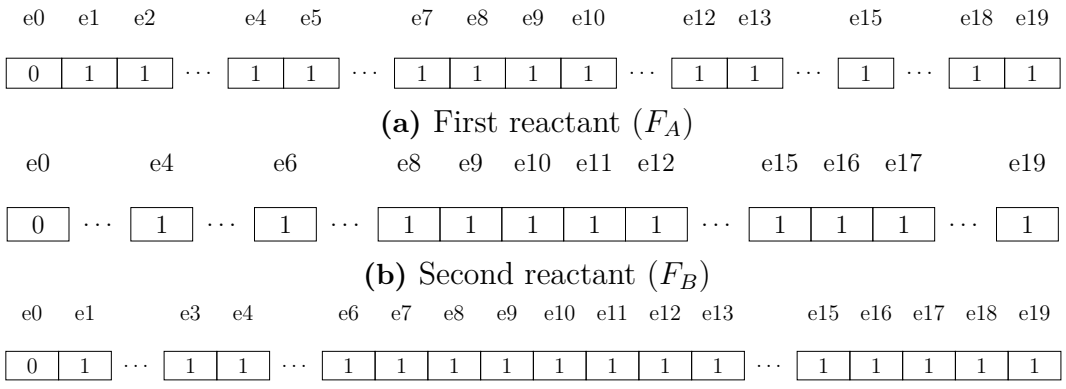
There are several ways to represent molecules for machine learning. One common representation is the use of molecular fingerprints, which can give information about both the structure and chemical properties. In fact, the most common measure on structural similarity between two molecules is molecular fingerprints [50]. Given the molecular fingerprints of the components in a reaction, one can describe the reaction with a reaction fingerprint.

2.7.1 Molecular Fingerprints

Molecular fingerprints are representations of chemical structures that encode features of molecules in formats that can be utilized by computers. Extended-connectivity fingerprints (ECFPs) are a popular fingerprint methodology that was explicitly designed to capture features relevant to how molecules affect other matter, e.g., the effects of drugs on living matter [51]. ECFPs are obtained by using a refined version of the Morgan algorithm [52]. As described in detail by [51], the ECFPs generation process has three sequential steps: (1) Each atom is assigned an integer, e.g., its atomic number, as an initial identifier. These initial identifiers are added to the fingerprint set of all identifiers; (2) A repeated (for a predetermined number of times for each atom) iterative step in which each atom updates its identifier by applying a hash function to an array containing its own identifier and the identifier of

its immediate neighboring atoms. In each iteration, identifiers containing duplicate information about the structure of a region of the molecule are removed. Moreover, the remaining new identifiers are added to the fingerprint set; (3) All duplicate identifiers are removed so that the set of all remaining identifiers uniquely define the fingerprint. However, this last step can be skipped in order to obtain an occurrence count of each identifier. This is then called a *fingerprint with counts*. When utilizing Macau with side information, (molecular) fingerprints with counts are used.

After each iteration for an atom in step (2), the update will create an identifier that represents larger and larger circular substructures around the atom. After the execution of the Morgan algorithm, each identifier (or the occurrence count of the identifier) is saved in the corresponding entry in a array with binary or count values. For instance, for a molecule with the identifier "123", the 123rd entry in the array is considered to have a non-zero value (either 1 or the occurrence count). Hence, the number of bits in the encoding is equal to the number of entries in the array, and the space of the hash function needs to fit the number of bits. As described by [51], different fingerprints are obtained by using different initial identifier rules and number of iterations in the update step. In fact, the standard ECFPs are obtained by using the ECFP rule for initialization; while the FCFP rule yields functional-class fingerprints (FCFPs), which are other popular fingerprints that are intended to capture "abstract role-based substructural features" [51]. To distinguish between the different number of iterations in step (2), the name convention FCFP4 and ECFP4 denotes the corresponding initialization by a four-character string (e.g., "FCFP" and "ECFP"); followed by a number (i.e., 4 in this example) representing the effective diameter of the largest feature, which is equal to twice the number of iterations.



(a) First reactant (F_A)

(b) Second reactant (F_B)

(c) Product (F_C)

Figure 2.5: 20-bits ECFP6 molecular fingerprints of the reactants and product for the reaction in Figure 1.3.

For the reaction in Figure 1.3, if utilizing a 20-bits ECFP6 fingerprint then the arrays in Figure 2.5 would correspond to the fingerprints of the first reactant (i.e., the aryl halide), second reactant and the product, respectively. The fact that there are ones at the 4th entry (e4) of each array means that fingerprint sets of all molecules includes the identifier "4" from the hash function. Note that only the entries with non-zero elements are shown, except for the first and last entries which are always

displayed. If fingerprints with counts were used, then the ones would be replaced by the occurrence counts. When utilizing the Merck data the fingerprints of the second reactant and product are not used since they are constant in the experiments. Then, the fingerprints (with counts) of all aryl halides (i.e., the different choices of the first reactant in the reaction in Figure 1.3) and the molecules corresponding to the reaction conditions were as side information. That is, all non-constant reaction components were encoded into fingerprints.

2.7.2 Reaction Fingerprints

Fingerprints can be used to describe reactions, known as reactions fingerprints. Let F_A, F_B and F_C be the molecular fingerprints of the first reactant, second reactant and the product, respectively, in a reaction. Then one way to represent a reaction fingerprint F_R is simply by $F_R = F_A + F_B - F_C$. This is how a reaction fingerprint is represented in this thesis. Bayesian neural networks utilizes reaction fingerprints (without counts). Figure 2.6 presents the 20-bits ECFP6 reaction fingerprint for a reaction with the molecular fingerprints in Figure 2.5.

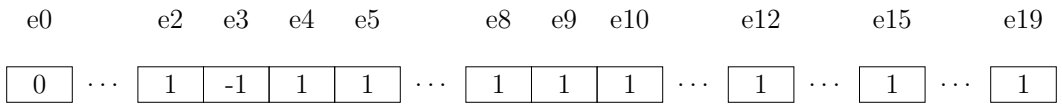


Figure 2.6: 20-bits ECFP6 reaction fingerprint (F_R) for the reaction in Figure 1.3. The molecular fingerprints for the reaction are given in Figure 2.5.

3
Methods

For the objective to explore the performance of active learning based on uncertainty quantification, this active learning approach was compared to expanding the training set without any information of the uncertainty. In particular, the variance of the predictions was utilized as uncertainty, and this approach was compared to random sampling and to methods for selecting data points with different reaction components. Recall that reaction components denote both the reaction conditions and the reactants of a reaction. Random sampling means that random points are added to the training set. The comparison of the different approaches to expand the training set was performed on the Pfizer and Merck data. For the objective to evaluate uncertainty quantification as a measure of the predictive performance of a neural network, Bayes by Backprop and MC dropout was utilized on the AZ ELN data. In particular, the theory in Section 2.2 was used to quantify the uncertainty in the output from these networks.

3.1 Active Learning Using Uncertainty Quantification

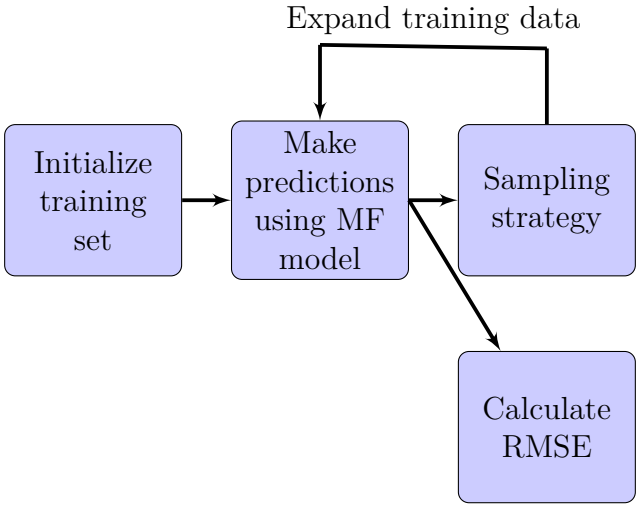


Figure 3.1: A flowchart showing the methodology for expanding the training set and predicting product yield using matrix factorization (MF).

The performance of active learning was investigated by predicting yields of the products of the Pfizer and Merck data. This is an forward synthesis prediction problem, see Figure 1.2b where some reaction components are held constant and some are varied. Moreover, only the yield of the product is of interest.

Figure 3.1 displays a simple flowchart showing the methodology that was used for expanding the training set to predict the observed (product) yields, which used the matrix factorization (MF) model Macau, see Section 2.5.2. The initialization utilized either random sampling or the Kennard-Stone algorithm. Three different sampling strategies were used: Random sampling, Uncertainty sampling and the Kennard-Stone algorithm (see Section 3.1.2 for more specifics). Uncertainty sampling had the option to use either absolute or relative uncertainty sampling and either with out without correlation simulations. See Section 3.1.3 for more information on the uncertainty sampling strategy, including details about the correlation simulations, that was used in this thesis.

With Random initialization, four comparisons were executed:

- (absolute) uncertainty without correlation simulations compared to relative uncertainty sampling without correlation simulations
- uncertainty sampling (with correlation simulations) compared to uncertainty sampling without correlation simulations
- uncertainty sampling without correlation simulations compared to random sampling
- uncertainty sampling without correlation simulations compared random sampling where all runs with Macau utilized fingerprints with counts as side information

Moreover, for initialization utilizing Kennard-Stone algorithm, one comparison was done:

- uncertainty sampling without correlation simulations compared to sampling utilizing Kennard-Stone algorithm

Note that "(absolute) uncertainty" and is used unless otherwise stated. The comparisons were performed on both the Pfizer and Merck data, except the runs utilizing either Kennard-Stone algorithm or fingerprints which were only compared on the Merck data. Also, to enable the use of fingerprints, 2048-bits ECFP6 fingerprints with counts were obtained from the chemical structures in the Merck data. The objective of using fingerprints was to investigate if, and in what way, additional chemical information would affect the comparison between uncertainty sampling without correlation simulations and random sampling. See Section 3.1.1 for more information on how the data was incorporated into the prediction model (Macau).

For all runs, the initial training data had a size of 90 unique data points, i.e., 90 unique permutations of the different reaction components (categorical variables). The size of 90 points was chosen since this is often the minimum number of different laboratory experiments that can be conducted in a single run. A 10-fold cross validation was utilized by splitting the Pfizer and Merck data into ten randomly shuffled subsets of (almost) equal size. Each subset was used as a test set once,

which generated ten runs for each sampling strategy where the other 9 subsets were available to be added to the training set. The run where the first subset was used as a dedicated test set is denoted as Fold 1 and so forth. The folds did always contain the same data so that the performance could be compared between all runs and sampling strategies. When utilizing random initialization, the initial training sets for each fold are the same for all sampling strategies in order to obtain an equal starting point, i.e., a fair comparison between sampling strategies. When utilizing Kennard-Stone algorithm as initialization, the initial training sets for each fold are randomly selected from possible sets obtained by Kennard-Stone algorithm. This was done in order to investigate the affects of different starting points.

After the initialization, Macau was used to predict the yields of the Pfizer and Merck data. In order to capture the variance in the predictions the yields were predicted ten times by utilizing Macua on the expanded training data. This was done after each sampling strategy had added 90 new points to the training set. For all runs with Macau, 400 burnins and 1600 samples were used when predicting the yields. This seems to give sufficient convergence in the samples. The predicted yield of each entry was the sample mean across all samples of the yield; while the (predictive) uncertainty of each entry/data point was the mean across all runs of the (unbiased) sample variance, which was calculated between all 1600 samples for each predicted entry. To obtain a scalar measure on the variance in the predictions, the predictive variability is here defined as the mean across all samples of the predictive uncertainty of each entry. Sampling was executed by adding 90 points in each iteration until all points, except the ones in the test set, had been added to the training test.

Using the predictions in each iteration, the performance of the expanded training set of all sampling strategies was evaluated by calculating the root-mean-square error (RMSE) on the dedicated test set for each run. The different sampling strategies were compared by calculating the differences of the mean RMSE (across all runs) and mean predictive variability (across all runs) for different numbers of points in the training set. Also, the 95% approximate confidence intervals of the differences of mean RMSE and mean predictive variability were calculated. These confidence intervals were calculated by assuming that predictions of different sampling strategies are independent. Furthermore, the predictions were used to determine the area under the precision-recall curve when a positive outcome (successfully reaction) corresponded to a (predicted) yield greater or equal to 5%. Recall that in order to determine the recall and precision, outcomes are divided into positive and negative outcomes.

To investigate the differences between the sampling strategies, the observed yield that was added in each step was stored. These were displayed by plotting the observed yields in each iteration, the plot uses a swarmplot on top of a boxplot. The box shows the upper, median and lower quartiles while the whiskers extend to show the rest of the points, except for the points that are determined to be outliers as a function of the range of the box (known as the interquartile range). Also, for the comparison between uncertainty sampling and random sampling, a Mann–Whitney U test was performed on the stored data in each step, in order to determine if the training sets from the different strategies were significantly different.

A Mann-Whitney U test [53] is a nonparametric test of the null hypothesis that the distributions of two random variables are equal. For sufficiently small p-values, the null hypothesis can be rejected. For the comparison between (asbolute) uncertainty sampling and relative uncertainty sampling, a single prediction was done by using the points that were added until iteration 16, which is where the RMSE and variance starts to converge according to the results in Section 4.1.1. This means that the model solely utilizes the points added up to iteration 16, and then performs a single prediction on each point in the test set. By using these predictions the following relative error was calculated and plotted against the observed (true) yield of the points in the test set

$$\text{relative error} = \frac{|y_{\text{pred}} - y_{\text{true}}|}{\frac{|y_{\text{pred}}|+|y_{\text{true}}|}{2}}, \tag{3.1}$$

where y_{pred} is the predicted yield and y_{true} is the observed (true) yield.

3.1.1 Data Implementation

As previously mentioned, both the Merck and Pfizer data were used for the different sampling strategies. Also, 2048-bits ECFP6 fingerprints with counts were used as side information to Macau when utilizing the Merck data. This section attempts to explain how this data was used by Macau.

As mentioned in Section 1.4.1, the Pfizer and Merck data consists of four and five different reaction components, respectively. The reaction components are denoted as categorical variables with different number of choices. Table 3.1 highlights the different number of choices for each reaction component. For both datasets, one data point corresponds to one choice of each reaction component (categorical variable) and the combination has an observed yield, which Macau predicts. That is, one data point from the Pfizer data consists of one choice of reactant 1, reactant 2, solvent, base and ligand. This generates in total 4608 data points for both the Pfizer and Merck data. However, note that the Merck data has no observed yield for 8 data points.

For the data to be utilized in Macau, the set of all permutations of choices of reaction components and corresponding yields was seen as a tensor. Each dimension of the tensor represents a reaction component, and the elements of each dimension represent the choices of the corresponding reaction component. An entry in the tensor is the yield corresponding to the elements of the dimension. For instance, for the Pfizer data with five reaction components, we have a tensor $Y_{i,j,k,l,m}$ of rank 5, i.e., five dimensions, where the choices of the indices i, j, k, l and m correspond to choices of the reaction components. Choosing the first reactant 1, second reactant 2, third solvent, fourth base and fifth ligand corresponds to entry $Y_{1,2,3,4,5}$ in the tensor, which is the yield corresponding to these choices of reaction components. Macau performs tensor factorization to predict the entire tensor using the known entries (observed yields) in the training data. The input to Macau is given as rows for each data point of the training data. Each row consist of the indices of the entry and its corresponding observed (true) yield.

The 2048-bits ECFP6 fingerprints with counts were obtained for the Merck data

by using the published chemical structures of the different choices of aryl halides, ligands, additives and bases. To incorporate the fingerprints as side information to Macau, the fingerprints were added as information to the corresponding elements of each rank of the tensor. E.g the fingerprint of the first aryl halide was added to element $Y_{1,.,.,.}$ of the tensor such that additional information about the structure of the first aryl halide is known.

Table 3.1: The number of possible choices of each reaction component of the Pfizer and Merck data.

(a) Pfizer data

Reaction component	# choices
Reactant 1	4
Reactant 2	3
Solvent	4
Base	8
Ligand	12

(b) Merck data

Reaction component	# choices
Aryl halide	16
Ligand	4
Additive	24
Base	3

3.1.2 Kennard-Stone Algorithm

In order to use the Kennard-Stone algorithm, a dissimilarity measure is needed in order to determine the distance between two combinations of choices of categorical values. Therefore, we used the count of different categorical values as the dissimilarity measure between two data points. This was done in order to ensure that all choices of reaction components are equally important and exploring different choices is of importance. For instance, for the data points corresponding to the entries $Y_{0,2,0,0,1}$ and $Y_{1,2,1,4,1}$, the distance is equal to $d = 1+0+1+1+0 = 3$. Unfortunately, this measure introduces a lot of ties when selecting new points. However, the idea is that it, hopefully, provides an experimental design for non-continuous levels that could be a feasible plan when conducting laboratory experiments. The ties were resolved by collecting all tied points and then randomly selecting a point from this collection. Hence, this should perform similarly to random sampling but with a stricter condition on what points can be used. The idea for this approach was to obtain a better variation between different reaction components in the training data.

3.1.3 Uncertainty Sampling

Uncertainty sampling was performed by iteratively selecting the uncorrelated points associated with the greatest uncertainty in the predictions, as seen in Algorithm 4. In particular, the predictive uncertainty (defined above as the sample variance) was utilized to assess the uncertainty of the predictions. Also, the option of using relative uncertainty was available. The relative uncertainty σ^2_{rel} of each prediction was calculated as

$$\sigma^2_{\text{rel}} = \frac{\sigma^2_{\text{pred}}}{1 + y_{\text{pred}}},$$

where σ^2_{pred} is the predictive (absolute) uncertainty of the entry/data point and y_{pred} is the predicted yield of the corresponding entry. The addition of one (1) in the denominator was utilized since $y_{\text{pred}} = 0$ is possible. Moreover, correlated points were only selected if no more uncorrelated points were available; for each iteration, where N new points were to be added, the list of correlated points was initially empty. Uncertainty sampling without correlation simulations did not consider the correlation between points and just added the points with the highest predictive uncertainty. In each iteration, when sampling N new points, a correlated point was defined as a point whose sample variance decreased below a certain threshold when the previous N_c points (or the number of available points if less than N_c points are available) were added to the training set. Checking which points that fulfill this definition, in order to find correlated points, are called correlation simulations. This was done in order to not add points that the model becomes less uncertain about when N_c new points are added to the training set. This was an attempt to consistently decrease the uncertainty when adding new points to the training set. That is, the usage of correlation simulations was based on the idea that it might be suboptimal to add multiple points which enclose the same information. For the correlation simulations, the predicted yields of these points were used as the desired output from Macau. Also, the threshold was determined by the mean predictive uncertainty of the N_c selected points times a constant. This constant was set to 0.8. Finally, the number of points to add to the training set before utilizing uncertainty sampling was set to $N_c = 5$ since it resulted in reasonable results and time complexity. That is, the approach in this thesis tries to simulate the correlations of a group of points, instead of simulating correlations for each point separately.

3.1.4 Difference in Yield Between Neighbours

Table 3.2: Average absolute difference in yield between points and its neighbours at different distances. The average differences are displayed with the same number of significant figures of the yields from the corresponding data set.

(a) Pfizer data

Distance	Average difference
1	17.40
2	24.30
3	29.23
4	33.11
5	36.39

(b) Merck data

Distance	Average difference
1	19
2	27
3	31
4	33

Matrix factorization predicts the data by finding relationships between the rows and columns (or corresponding entities in the higher-dimensional cases). Thus, the data should contain such relationships if Macau is able to make reasonable predictions. By using the distance introduced for the Kennard-Stone algorithm in Section 3.1.2, we calculated the average absolute difference in yield between a point and its neighbours at a specific distance. This was done in for both the Pfizer and Merck data which gave the average differences presented in Table 3.2. The distances are displayed with

the same precision as provided in the corresponding data. Recall that the distance between two data points is the number of different choices of reaction components.

Algorithm 4 Uncertainty sampling

Input: Current training set S_c, P_c prediction of all of all entries/data points with current training set, predictive uncertainties V_c of all entries/data points with current training set, number of points N (dynamic variables that depends on the number of available points) to be added to the current training set, number of points $N_c = 5$ to add before simulating correlations, set S of all points points except those in the test set S_T.
Initialize empty list of correlated points $L_{\text{corr}} := \emptyset$, new training set $S_n := S_c$ and new points $P_{\text{new}} := \emptyset$
while $|S_n \setminus S_c| < N$ **do**
  Let $N_{\text{corr}} := \min\{|S \setminus S_n|, N_c\}$
  for $1, \ldots, N_{\text{corr}}$ **do**
    if $S \setminus \{S_c \cup L_{\text{corr}}\} \neq \emptyset$ **then**
      Let new data point $P_{\text{new}} = P_{\text{new}} \cup \arg\max_{i \in S \setminus \{S_c \cup L_{\text{corr}}\}} V_c(i)$
    else
      Let new data point $P_{\text{new}} = P_{\text{new}} \cup \arg\max_{i \in S \setminus \{S_c\}} V_c(i)$
    end if
  end for
  $S_n = S_n \cup P_{\text{new}}$
  if utilizing correlation simulations **then**
    for $P \in S$ **do**
      Make prediction for P with $S_c \cup P_{\text{new}}$ as training set, using predicted values $P_c(\{P_{\text{new}}\})$ as the temporary "true" output of the points in P_{new}
      Calculate new predictive uncertainties (variances) $V_{\text{new}}(P)$
    end for
    Let $V_{P_{\text{new}}} := \frac{1}{|P_{\text{new}}|} \sum_{P \in P_{\text{new}}} |V_{\text{new}}(P)|$
    for all points $P \in S \setminus S_n$ **do**
      if $V_c(P) - V_{\text{new}}(P) \geq V_{P_{\text{new}}} \cdot C$ **then**
        $L_{\text{corr}} = L_{\text{corr}} \cup P$
      end if
    end for
  end if
end while
Output: New training set S_n

3.2 Evaluating Uncertainty in Neural Networks

3.2.1 Bayes by Backprop

The variational inference method Bayes by Backprop was used to learn a weight distribution in a Bayesian neural network, Section 2.4.6.1. This means that the true posterior $\pi(\boldsymbol{w}|\mathcal{D})$ was approximated with the variational distribution $q(\boldsymbol{w}|\theta)$, which is parameterized by θ. The objective was thereby to learn the optimal parameters of the variational distribution, Equation 2.6. The variational distribution was assumed to be a diagonal Gaussian with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma^2}$. To ensure that the variance is non-negative, we used the pointwise transformation $\sigma = \log(1 + \exp(\rho))$ suggested by [30]. Thus, the variational parameters are $\theta = (\boldsymbol{\mu}, \boldsymbol{\rho})$. With a diagonal Gaussian, the reparameterization trick in Section 2.4.6.1 takes the form

$$\boldsymbol{w} = t(\theta, \epsilon) = \boldsymbol{\mu} + \log(1 + \exp(\boldsymbol{\rho})) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{3.2}$$

This yields samples from $q(\boldsymbol{w}|\theta)$ by sampling noise ϵ that does not depend on the sought parameters θ.

The loss function is given by a slightly modified version of Equation (2.22),

$$\tilde{\mathcal{F}}(\mathcal{D}, \theta) = \frac{\beta}{T} \sum_{t=1}^{T} \log \left(\frac{q(\boldsymbol{w}^{(t)}|\theta)}{\pi(\boldsymbol{w}^{(t)})} \right) + \frac{1}{T} \sum_{t=1}^{T} \log \left(\pi(\mathcal{D}|\boldsymbol{w}^{(t)}) \right), \tag{3.3}$$

where T is the number of samples and $t(\theta, \epsilon)$ is defined as in Equation (3.2). The parameter $\beta \in [0, 1]$ reduces the effect of the complexity loss. This means that the model will train more to fit the data than to remain close to the prior. Note that $\beta = 1.0$ yields the loss in Equation (2.22). Finally, the distribution parameters θ are learnt through backpropagation of Equation (3.3) w.r.t. θ. Furthermore, the BNN was implemented using PyTorch 1.4 with CUDA 10.1. This allowed for the easy use of the automated differentiation inherent to PyTorch tensors, allowing for easy forward- and backward passes in the network. Since the goal is to classify the correct ligand, the final layer uses a softmax function.

The terms in Equation (3.3) can be computed as follows; since the variational posterior was a diagonal Gaussian, it can be expressed as the product over each element

$$q(\boldsymbol{w}^{(t)}|\theta) = \prod_{i \in I} \mathcal{N}(w_i^{(t)}|\mu_i, \sigma_i^2),$$

where $w_i^{(t)}$ is the $i-$th weight of the $t-$th sample and I is the indexing set for the weights. The loss requires the logarithm of $q(\boldsymbol{w}^{(t)}|\theta)$, which takes the form

$$\log\Big(q(\boldsymbol{w}^{(t)}|\theta)\Big) = \sum_{i \in I} \log\Big(\mathcal{N}(w_i^{(t)}|\mu_i, \sigma_i^2)\Big),$$

and this term can easily be computed since the mean and variance is known. The prior was assumed to also be a diagonal Gaussian but with mean 0 and variance σ_p^2. The log-prior is thereby given by

$$\log\Big(\pi(\boldsymbol{w}^{(t)})\Big) = \sum_{i \in I} \log\Big(\mathcal{N}(w_i^{(t)}|\mu_i, \sigma_p^2)\Big).$$

Furthermore, the log-likelihood loss, $\log\left(\pi(\mathcal{D}|\boldsymbol{w}^{(t)})\right)$, in Equation (2.22), was computed with PyTorch's negative log-likelihood loss of the output given the weight sample $\boldsymbol{w}^{(t)}$. These terms constitute a single sample of the loss. The parameters are learnt by backpropagating the obtained loss from multiple such samples.

3.2.1.1 MNIST Performance

In order to validate that the Bayes by Backprop implementation was correct the performance of the network on the MNIST data set was compared to the authors' performance [30]. The MNIST data set consists of 70 000 pixel images of size 28 by 28 pixels. Each data point consists of a grayscale image of a handwritten digit in the range 0-9, and the corresponding label. This set is further split into a training and validation set of 60 000 and 10 000 images, respectively. To have a fair comparison the network consisted of an input layer with 784 neurons and two hidden layers, both consisted of 1200 neurons, and finally an output layer with 10 neurons. The ReLU activation function was used for all layers except for the last where instead the softmax activation function was utilized.

The prior distribution on the weights was assumed to be $\mathcal{N}(0, 0.1^2)$. The loss, Equation (3.3), with $\beta = 1.0$ was approximated with $T = 3$ samples. The optimization of the network was performed by SGD with a learning rate of 10^{-3}, Algorithm 3, and used a mini-batch size of 128. The mean μ was initialized according to

$$\mu_i \sim \mathcal{U}\left(-0.2, 0.2\right),$$

and ρ according to

$$\rho_i \sim \mathcal{U}\left(-3, -2\right).$$

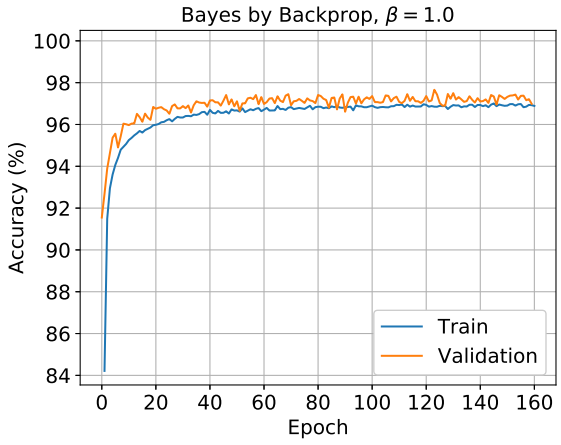


Figure 3.2: The MNIST validation accuracy of our PyTorch implementation of Bayes by Backprop. The obtained accuracy is similar to that obtained by the authors of the method [30].

Figure 3.2 illustrates the accuracy on the training and validation set for our model. Note that their trends are very similar. However, the validation set is more noisy. This can probably be attributed to the fact that the validation set is unknown to

the model and that the validation set only utilizes one stochastic forward pass, while the training averages over $T = 3$ stochastic forward passes. This might also explain why the validation accuracy is slightly higher, but this can also be a consequence of the fact that this approach learns more robust patterns [30]. The result is also similar to that obtained by [30], and the discrepancy might be explained by the fact that the exact hyperparameters of their implementation are unknown. This result indicates that our implementation could be correct.

3.2.2 AZ ELN Implementation

The objective of the network is to predict which ligand that was used in a reaction, given the reaction as an input. This means that the network performs a variant of retrosynthesis prediction, see Figure 1.2a, where only the used ligand is predicted, while both the reactants and the product are known. That is, the model predicts one of the reaction conditions. In total, 13 different ligands have been used in the reactions, and Table 3.3 presents how often each ligand is used. The reactions in the data are encoded as 2048-bits ECFP6 reaction fingerprints F_R, which are described in Section 2.7.2. This means that the input $\boldsymbol{x}$ to the network is F_R. The network was implemented with two hidden layers, which used ReLU as activation function, and an output layer which used softmax in order to obtain the predicted ligand. The final layer consisted of 13 neurons, where each neuron corresponded to a ligand.

Table 3.3: How often each ligand is present in a reaction. Note that the data is severely imbalanced.

Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12
# observations	250	13	947	143	14	13	6	14	96	1964	64	19	73

The data was split into a test set, which consisted of 10% of the observations, and a training set and a validation set. The validation and training set was obtained with 5-fold CV on the remaining 90% of the data. Both the CV and test set was partitioned with stratified sampling, due to the large class imbalance. This ensured that each fold had roughly the same class balance. The loss and top n accuracy was computed for all three sets. Top n accuracy measures how often the true ligand is present in the n most probable predictions. The diversity in the predicted ligands, i.e., the number of predicted ligands, was also computed.

3.2.2.1 Bayes by Backprop

The hidden layers in this implementation consisted of 64 neurons each. The prior distribution of the weights was assumed to be Laplace$(0, 0.1)$. The loss, Equation (3.3), with $\beta \in \{0.1, 0.5, 1.0\}$ was approximated with $T = 10$ samples. This loss was optimized with Adam [54] with a learning rate of 10^{-3} and used a mini-batch size of 128. The mean μ of the variational posterior was initialized according

to the Xavier uniform initialization

$$\mu_i \sim \mathcal{U}\left(-\sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}, \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}\right),$$

where "fan_in" and "fan_out" is the number of inputs and outputs from the node, respectively. ρ was initialized as

$$\rho_i = \log\left(\exp\left(\sqrt{\frac{2}{\text{fan\_in}}} - 1\right)\right).$$

In order to evaluate how many ligands the model manages to predict each reaction was passed through the network 100 times, and the prediction of each pass was stored. Thereafter the most frequent observation for each reaction was deemed as the top prediction. The predictive diversity is simply the number of unique such top predictions present in the input set. An important note is that the predictive diversity does not illustrate how many of the predicted ligands that are predicted correctly.

Table 3.4: Hyperparameters used in the Bayes by Backprop implementation.

Layer1	Layer 2	Learning rate	Batch size	Prior	T	Epochs
64	64	10^{-3}	128	Laplace(0, 0.1)	10	1000

3.2.2.2 MC Dropout

The hidden layers in this implementation consisted of 128 neurons each. This ensured that the network almost had the same number of trainable parameters as the Bayes by Backprop implementation since that implementation trains both mean and variance. The weights are initialized according to Xavier uniform initialization

$$\boldsymbol{w}^{(i)} \sim \mathcal{U}\left(-\sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}, \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}\right),$$

where "fan_in" and "fan_out" is the number of inputs and outputs from the node, respectively.

The likelihood loss, $\log\Big(\pi(\mathcal{D}|\boldsymbol{w}^{(t)})\Big)$, in Equation (2.24) was computed with PyTorch's negative log-likelihood loss of the output given the weight sample $\boldsymbol{w}^{(t)}$. The loss in Equation (2.24) was optimized with Adam [54] with an initial learning rate of 10^{-4} and with a mini-batch size of 64. The model uses a constant dropout rate of either 0.2 or 0.5 for both training and testing.

In order to evaluate how many ligands the model manages to find each reaction was passed through the network 100 times, and the prediction of each pass was stored. Thereafter the most frequent observation was deemed as the top prediction. The predictive diversity is simply the number of unique such top predictions present in the input set. An important note is that the predictive diversity does not illustrate how many of the predicted ligands that are predicted correctly.

Table 3.5: Hyperparameters used in the MC dropout implementation.

Layer 1	Layer 2	Learning rate	Batch size	Prior	Epochs
128	128	10^{-4}	64	Normal(0, 0.1^2)	1000

3.2.3 Uncertainty Estimation

An advantage of representing weights by distributions, instead of point estimates, is that one can estimate the posterior predictive distribution, Equation (2.3), for inference of possible outcomes $\boldsymbol{y}^*$ given new input data $\boldsymbol{x}^*$. The variance of this distribution can be used to quantify the uncertainty of a prediction, Section 2.2. Equation (2.13) gives an estimate of the uncertainty in a variational inference setup. Therefore we need to compute the expectations $\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*]$ and $\mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*]$.

Let $\hat{p}^{(t)}(\boldsymbol{x}^*)$ denote the softmax output from the network with weights $\boldsymbol{w}^{(t)}$ and input $\boldsymbol{x}^*$ for the MC sample t. Since the last layer of our network is a softmax-layer the resulting output can be interpreted as probabilities. More concrete, each element of $\hat{p}^{(t)}(\boldsymbol{x}^*)$ corresponds to the probability that the network would predict the corresponding class. Thereby, $\hat{p}^{(t)}(\boldsymbol{x}^*)$ can be used to approximate

$$\mathbb{E}_{\pi(\boldsymbol{y}^*|\boldsymbol{x}^*,\boldsymbol{w}^{(t)})}[\boldsymbol{y}^*] \approx \hat{p}^{(t)}(\boldsymbol{x}^*),$$

and $\overline{p}(\boldsymbol{x}^*) = \frac{1}{T}\sum_{t=1}^{T}\hat{p}^{(t)}(\boldsymbol{x}^*)$, which is the approximation of the variational predictive distribution, can approximate

$$\mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*,\theta)}[\boldsymbol{y}^*] \approx \overline{p}(\boldsymbol{x}^*),$$

see Section 2.2. These two estimations thereby allow us to quantify the uncertainty in the predictions according to Equation (2.13). The key reason for this is that the output of the network can be interpreted as a probability. It is further worth noting that this approach is not limited to a softmax layer. In fact, any layer that outputs a probability distribution will suffice. The uncertainty was evaluated on the test set with $T = 1000$ stochastic forward passes.

We opted to decompose the uncertainty according [33], rather than the decomposition suggested by [31]. The main reason for this decision was that [33] pointed out that the decomposition suggested by [31] is not suitable for classification problems since it does not model the variance of the predictive probabilities. Furthermore, the approach by [33] does not require additional nodes at the final layer, thereby reducing the size of the final layer by a factor of two compared to the approach suggested by [31].

An additional approach utilized to evaluate the uncertainty was to pass an input through the network T times and for each forward pass save the predicted ligand, i.e., the ligand that corresponds to the maximum softmax entry. The algorithm is presented in Algorithm 5, and this uncertainty estimate will hereafter be referred to as "Naïve uncertainty". Since this approach only uses the actual predicted outcome of the network it contains less information than the decomposition discussed in Section 2.2. However, this approach provides an easy method to estimate the

uncertainty, and the estimated uncertainty is easy to interpret since it consists of a count of how many times that each ligand was deemed as the most probable one.

Algorithm 5 Naïve uncertainty

1: Let $\boldsymbol{x}^*$ be the input and let $T \in \mathbb{N}$ be the number of forward passes of $\boldsymbol{x}^*$ through the network. Furthermore, let $\hat{p}^{(t)}(\boldsymbol{x}^*)$ denote the softmax output from the network with input $\boldsymbol{x}^*$ and weight sample $\boldsymbol{w}^{(t)}$. Lastly, let possible_ligands be a list of the possible outcomes from the network
2: **for** $t = 1, \ldots, T$ **do**
3: predicted_ligand = $\arg\max \hat{p}^{(t)}(\boldsymbol{x}^*)$
4: possible_ligands[predicted_ligand]+ = 1
5: **end for**

Lastly, the entropy of the variational predictive

$$H(q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta)) \approx -\sum_{k=1}^{K} \overline{p}(\hat{\boldsymbol{y}} = e_k|\boldsymbol{x}^*) \log\left(\overline{p}(\hat{\boldsymbol{y}} = e_k|\boldsymbol{x}^*)\right),$$

was also used to evaluate the uncertainty of an input $\boldsymbol{x}^*$. This uncertainty estimate yields a scalar value for each input; this is to be compared to the decomposition which yields a matrix and the naïve uncertainty which yields a vector. A lower value of entropy indicates that the variational predictive distribution does not contain much information, which can be interpreted as a certain model.

4

Results

This chapter shows the results of the matrix factorization approach utilizing Macau for predicting yields and the Bayesian neural network for suggesting reaction conditions.

4.1 Active Learning Using Uncertainty Quantification

This section shows the results of the prediction of yields for different choices of reaction components when using different strategies to expand the training set. Recall that reaction components denote both the reaction conditions and the reactants of a reaction. Five comparisons, between sampling strategies, obtained on the Merck and Pfizer data are presented:

1. (absolute) uncertainty sampling without correlation simulations compared to relative uncertainty sampling without correlation simulations
2. uncertainty sampling compared to uncertainty sampling without correlation simulations
3. uncertainty sampling without correlation simulations compared to random sampling
4. Kennard-Stone Initialization: uncertainty sampling without correlation simulations compared to sampling using Kennard-Stone algorithm when both strategies were initialized using Kennard-Stone algorithm
5. uncertainty sampling without correlation simulations vs random sampling when both strategies utilized fingerprints as side information to Macau

Comparisons 1, 2, 3 and 5 used random initialization. Recall that "(absolute) uncertainty" is used unless otherwise stated.

4.1.1 (Absolute) Uncertainty Compared to Relative Uncertainty

This section presents the result of the comparison between (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations obtained on the Merck data. The corresponding comparison

obtained on the Pfizer data is presented in Appendix A.1.1.1. The results obtained on the Pfizer data are similar to the results that were obtained on the Merck data.

4.1.1.1 Merck Data

Figure 4.1 shows the difference

$$\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{relative uncertainty}}$$

of the mean RMSEs between (absolute) uncertainty sampling without correlation simulations ($\mu_{\text{RMSE}}^{\text{uncertainty}}$) and relative uncertainty sampling without correlation simulations ($\mu_{\text{RMSE}}^{\text{relative uncertainty}}$). The error bars display the 95% approximate confidence interval. Only the results from fold 1 and 2 of the Merck data are presented here, but the other CV folds show similar behaviour. These results show that (absolute) uncertainty sampling without correlation simulations yields a significantly lower RMSE compared to relative uncertainty sampling without correlation simulations since the difference $\Delta\mu_{\text{RMSE}}$ is negative (expect for the first and last iteration). Figure 4.3a shows the average difference over all folds. This result is consistent with the results of fold 1 and 2. That $\Delta\mu_{\text{RMSE}}$ initially is positive, see Figures 4.1 and 4.3a, illustrates that when the training set is small, then relative uncertainty without correlation simulations performs better than (absolute) uncertainty sampling without correlation simulations. However, the fact that $\Delta\mu_{\text{RMSE}}$ is negative when the training set increases illustrates that (absolute) uncertainty sampling without correlation simulations performs better for larger training sets.

Figure 4.2 shows the difference

$$\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty}} - \mu_{\text{predictive variability}}^{\text{relative uncertainty}}$$

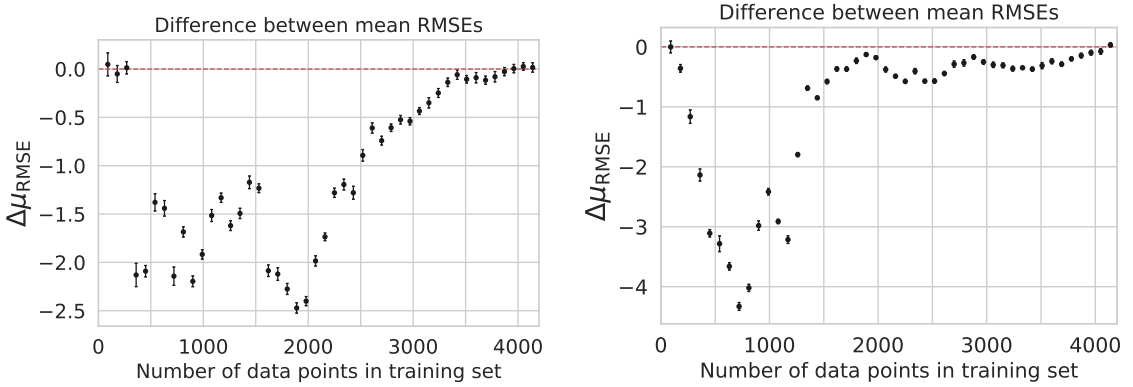
of the mean predictive variability between (absolute) uncertainty sampling without correlation simulation ($\mu_{\text{predictive variability}}^{\text{uncertainty}}$) and relative uncertainty sampling without correlation simulations ($\mu_{\text{predictive variability}}^{\text{relative uncertainty}}$). The error bars display the 95% approximate confidence interval. Only the results for fold 1 and 2 of the Merck data are presented here, but the other folds illustrated similar behaviour. Moreover, Figure 4.3b visualizes the average difference over all folds. These results indicates that (absolute) uncertainty sampling without correlation simulations yields lower sample variances of the entries compared to relative uncertainty sampling without correlation simulations since the difference is negative except for a few iterations in the beginning. Recall that in each iteration is 90 new points added to the training set. That $\Delta\mu_{\text{predictive variability}}$ mostly is negative indicates that (absolute) uncertainty sampling without correlation simulations makes more certain predictions, compared to relative uncertainty sampling without correlation simulations, when the size of the training set increases.

Figure 4.4 shows the area under the precision-recall curve (PR AUC) score of (absolute) uncertainty sampling without correlation simulation and relative uncertainty sampling without correlation simulations when utilized on fold 1 and 2 of the Merck data. Recall that a successful reaction (positive outcome in terms of the precision and recall) is defined as a reaction with a yield greater or equal to 5%. For fold

1, (absolute) uncertainty sampling seems to show an overall better PR AUC score compared to relative uncertainty sampling. However, for fold 2, the strategy with the best PR AUC score varies between each iteration. Also, relative uncertainty sampling seems to show an overall better PR AUC score in some folds not displayed here. Hence, the best scoring strategy seems to vary between folds, but both strategies seem to show a reasonably high score when there is a sufficient number of data points in the training set. This number seems to be similar for both strategies.

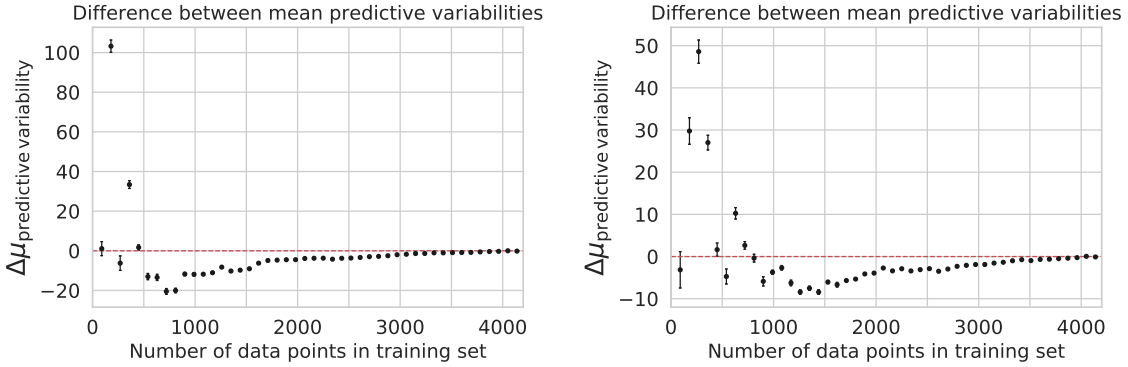
Figures 4.5a and 4.5b present the observed yield of the points that were added in each iteration for (absolute) uncertainty sampling without correlation simulation and relative uncertainty sampling without correlation simulations, respectively. This is visualized by utilizing a swarmplot on top of a boxplot. (Absolute) uncertainty sampling adds mainly points with low observed yield in the last iterations. Relative uncertainty sampling seems to add points with low observed yield in the beginning and only a few points with low observed yield in the end.

Figure 4.6 shows the observed yield plotted against the relative error, see Equation (3.1), of (absolute) uncertainty sampling and relative uncertainty sampling utilized on fold 1 and 2 of the Merck data. The relative error was calculated on the training points that have been added up till the 16-th iteration of the different strategies. The fact that the predictions from (absolute) uncertainty sampling have more points with a relative error of 2 illustrates that this strategy generates more predictions with 0% yield, compared to relative uncertainty. (Absolute) uncertainty sampling also seems to demonstrate more points with observed yields greater than 20% that obtains a relative error lower than 0.25. Relative uncertainty display more points with observed yields lower than 20% that obtains a relative error lower than 0.25.



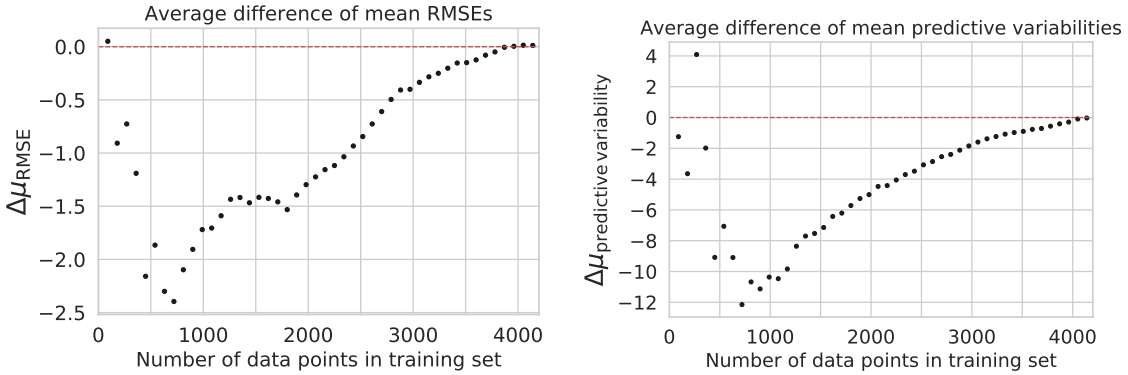
(a) Fold 1 **(b)** Fold 2

Figure 4.1: The difference $\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{relative uncertainty}}$ of the means of the RMSE between (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations as a function of the number of points from the Merck data added to the training set. The means were calculated from ten predictions. Random initialization was utilized, no correlation simulation was applied. Fold 1 and 2, of in total ten folds, of the cross validation are displayed. The red dotted line displays where the difference is equal to zero and the error bars show the 95% approximate confidence interval.



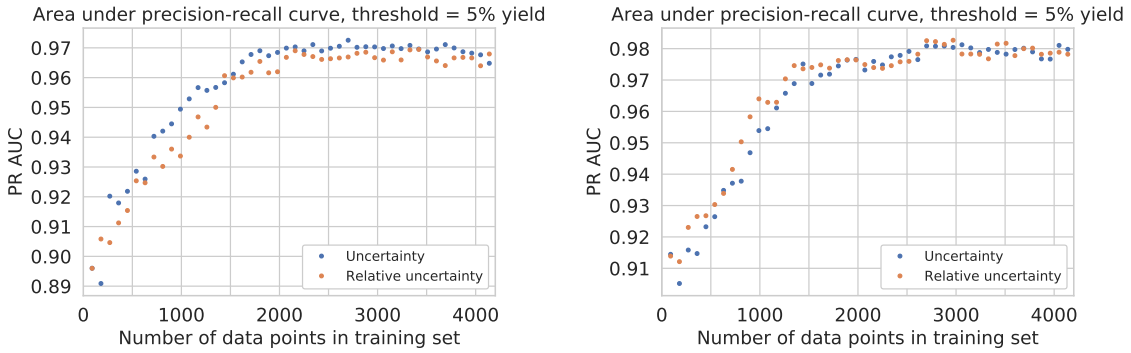
(a) Fold 1 **(b)** Fold 2

Figure 4.2: The difference $\Delta\mu_{\text{predictive variability}} = \mu^{\text{uncertainty}}_{\text{predictive variability}} - \mu^{\text{relative uncertainty}}_{\text{predictive variability}}$ of mean predictive variability of (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations as a function of the number of points from the Merck data that have been added to the training set. Fold 1 and 2, of in total ten folds, of the cross validation are displayed. The red dotted line displays where the difference is equal to zero and the error bars show the 95% approximate confidence interval.



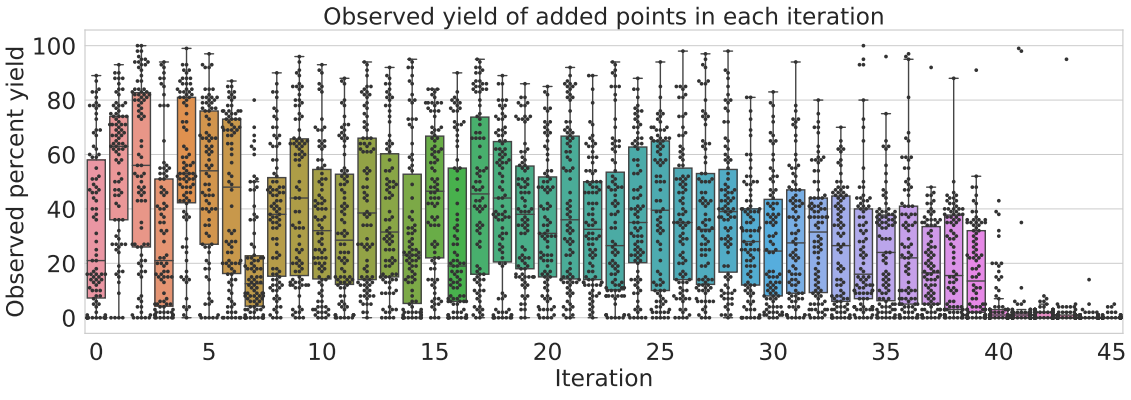
(a) RMSE **(b)** Predictive variability

Figure 4.3: The average differences $\Delta\mu_{\text{RMSE}}$ and $\Delta\mu_{\text{predictive variability}}$ of (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations over all folds as a function of the number of points from the Merck data that have been added to the training set. The red dotted line displays where the difference is equal to zero and the error bars show the 95% approximate confidence interval.

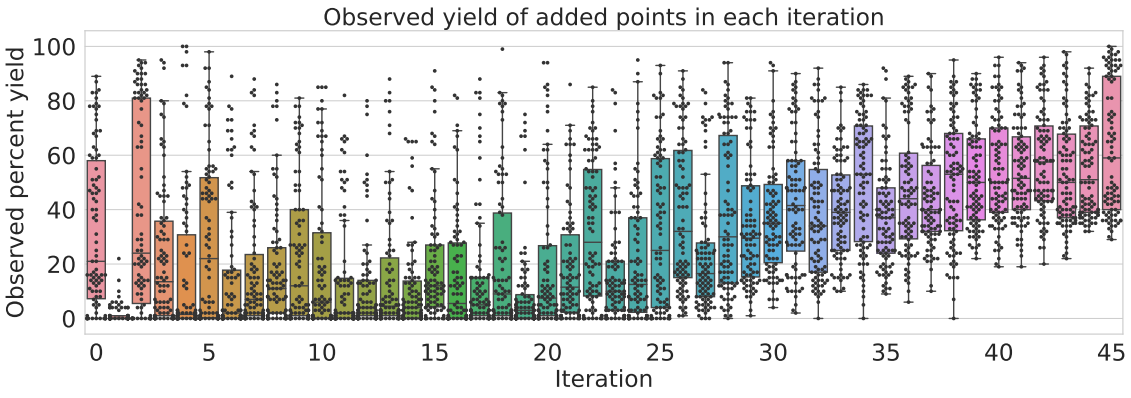


(a) Fold 1 **(b)** Fold 2

Figure 4.4: Area under precision-recall curve of (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations on the Merck data.



(a) (Absolute) uncertainty sampling

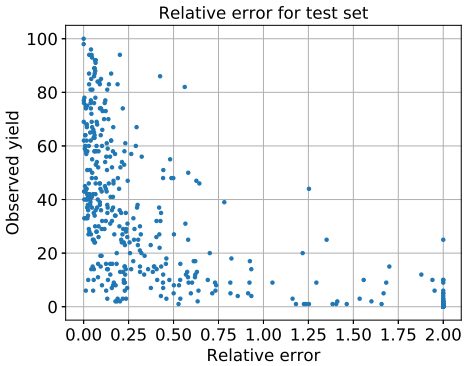


(b) Relative uncertainty sampling

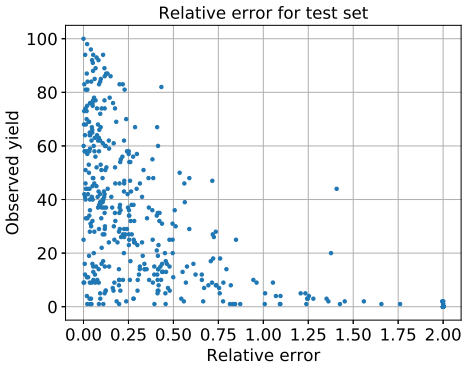
Figure 4.5: Observed (true) percent yield of points from the Merck data that have been added to the training set in each iteration when utilizing (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations. Only fold 1 is displayed but the other folds show similar behaviors. The two methods are initialized with the same randomly sampled points.

(a) Fold 1: (Absolute) uncertainty sampling

(b) Fold 1: Relative uncertainty sampling



(c) Fold 2: (Absolute) uncertainty sampling



(d) Fold 2: Relative uncertainty sampling

Figure 4.6: Observed yield plotted against relative error of the test set when utilizing (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations applied on the Merck data. Only fold 1 and 2 are displayed but the other folds shows similar behaviors. Macau was trained by using points that have been added to the training set up till the 16-th iteration. That is, in total 1530 points, of the different sampling strategies.

4.1.2 Uncertainty With and Without Correlation Simulations

This section presents the results of the comparison between uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations obtained on the Pfizer data. The corresponding results obtained on the Merck data is presented Appendix A.1.2.1. The results obtained on the Merck data are similar to the results shown for the Pfizer data.

4.1.2.1 Pfizer Data

Figure 4.7 shows the difference

$$\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty with correlation}} - \mu_{\text{RMSE}}^{\text{uncertainty without correlation}}$$

of the mean RMSE between uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations obtained on fold 1 and 2, of in total ten folds, of the cross validation. Figures 4.7a and 4.7b display these results for fold 1 and fold 2, respectively. The mean RMSEs were calculated from ten predictions and the error bars display the 95% approximate confidence interval over these predictions. Both sampling strategies were initialized with a random sample of size 90 of the available data points, i.e., all data points except the ones in the dedicated test set. For fold 1, as seen in Figure 4.7a, the strategy with the lowest RMSE seems to be uncertainty sampling with correlation simulations until the 9th iteration (where the first iteration is the random initialization and is denoted as the 0th iteration), where 900 points have been added to the training set, when uncertainty sampling without correlation simulations starts to yield a lower RMSE compared to uncertainty sampling with correlation simulations. Until the 38th iteration, where 3510 points have been added to the training set, uncertainty sampling without correlation simulations seems to yield a lower RMSE for the majority of the iterations but the difference is close to zero with the majority of the differences taking on values between $0.25 < \Delta\mu_{\text{RMSE}} < 0$.

For fold 2, as seen in Figure 4.7b, in the beginning, when a small number of points have been added to the training set, uncertainty sampling with correlation simulations seems to yield a lower RMSE. From the 14th iteration, where 1350 points have been added to the training set, the best performing strategy in regard to RMSE varies, although the performance is similar for the two methods. Figure 4.9a shows the average (over all folds) difference of mean RMSEs. This figure seems to indicate that uncertainty sampling with correlation simulations yields a lower RMSE compared to uncertainty sampling without correlation simulations until around 2000 points have been added to the training set, but the difference is close to zero for sufficient number of points in the training set. After this point, the strategy which yields lowest RMSE varies and, hence, it seems that no strategy performs significantly better than the other.

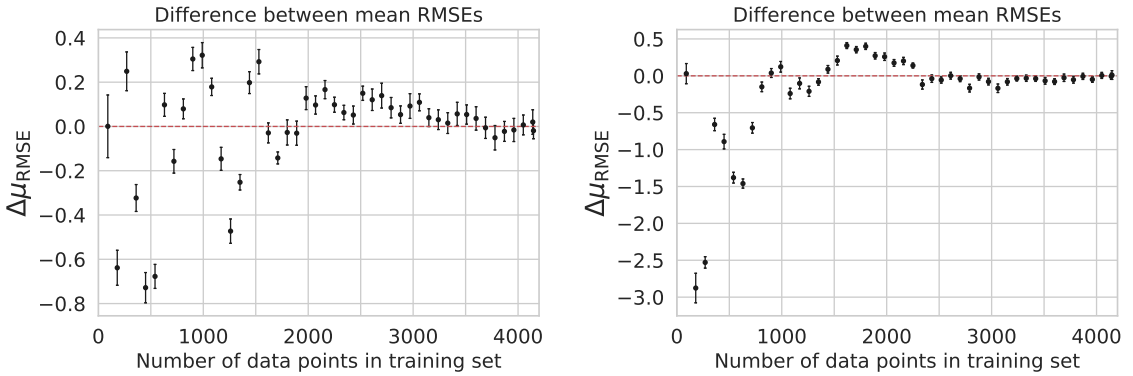
Figures 4.8a and 4.8b show the difference

$$\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty with correlation}} - \mu_{\text{predictive variability}}^{\text{uncertainty without correlation}}$$

of the mean predictive variabilities between uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations obtained on fold 1 and 2, respectively. The difference is displayed as a function of the number of added points to the training set. The error bars display the 95% approximate confidence interval across the multiple predictions.

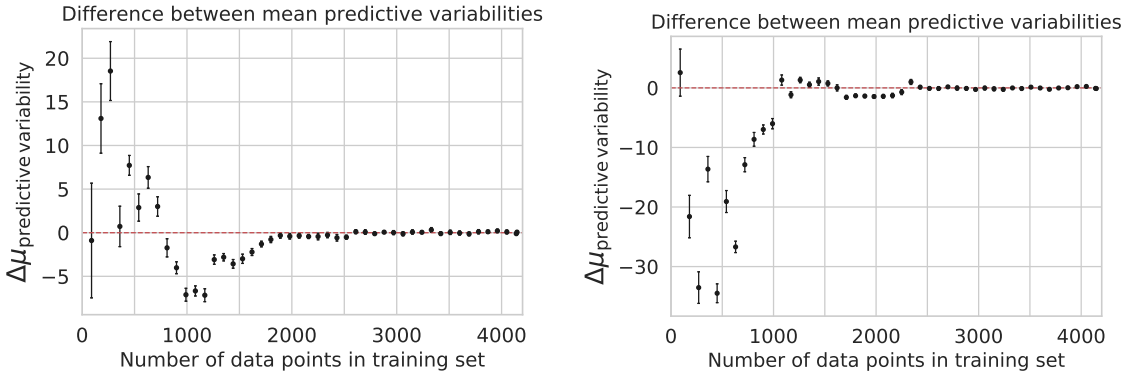
For fold 1, as seen in Figure 4.8a, uncertainty sampling without correlation simulations seems to yield a lower yield in the beginning but the range of the confidence interval is larger compared to the subsequent iterations. From the 8th iteration (where the first iteration is the random initialization and is denoted as the 0th iteration), where 810 points have been added to the training set, uncertainty sampling with correlation simulations starts to yield lower variability compared to uncertainty sampling without correlation simulations. However, starting at around the 20th iteration, where 1890 points have been added to the training set, the difference seems to be close to zero but still mostly negative. For fold 2, as seen in Figure 4.8b, uncertainty sampling with correlation simulations seems to give the lowest variability until the 10th iteration, where 990 points have been added to the training set. From the 10th iteration, uncertainty sampling without correlation simulations seems to yield the lowest predictive variability until iteration 18 when the difference is close to zero and starts to vary between being negative and positive. Figure 4.9b shows the average (over all folds) difference of mean predictive variability. This figure indicates that the average difference is mostly negative, i.e., uncertainty sampling with correlation simulations seems to yield a lower variance, but the difference is close to zero for a sufficient number of points in the training set. This illustrates that both methods are equally consistent between iterations, i.e., both methods show similar uncertainty in their predictions. Thus, neither method is superior to the other.

Figure 4.10 show area under the precision-recall curve (PR AUC) for fold 1 and 2 when a successful reaction (positive outcome) is defined as a reaction with a yield greater or equal to 5%. The performance seems similar, and the best strategy regarding PR AUC varies between each iteration. Figures 4.11a and 4.11b display the added observed yield in each iteration for uncertainty sampling without correlation simulations and uncertainty sampling with correlation simulations, respectively, utilized on fold 1. Both strategies show a similar behavior, where at the last iterations (from iteration 39 to iteration 45) the majority of the added points has yield below 30%.



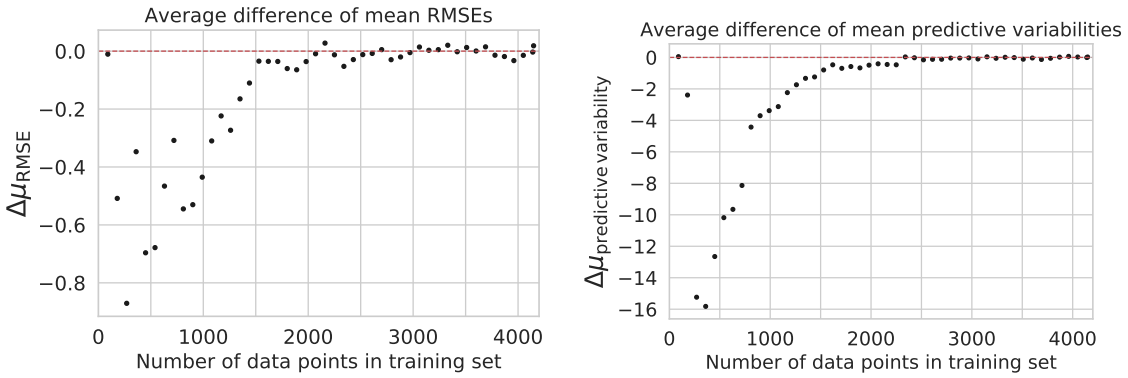
(a) Fold 1 **(b)** Fold 2

Figure 4.7: The difference $\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{uncertainty without correlation}}$ of the mean RMSEs between uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations as a function of the number of points from the Pfizer data that have been added to the training set. Fold 1 and 2, of in total ten folds, of the cross validation are displayed. The red dotted line displays where the difference is equal to zero and the error bars show the 95% approximate confidence interval.



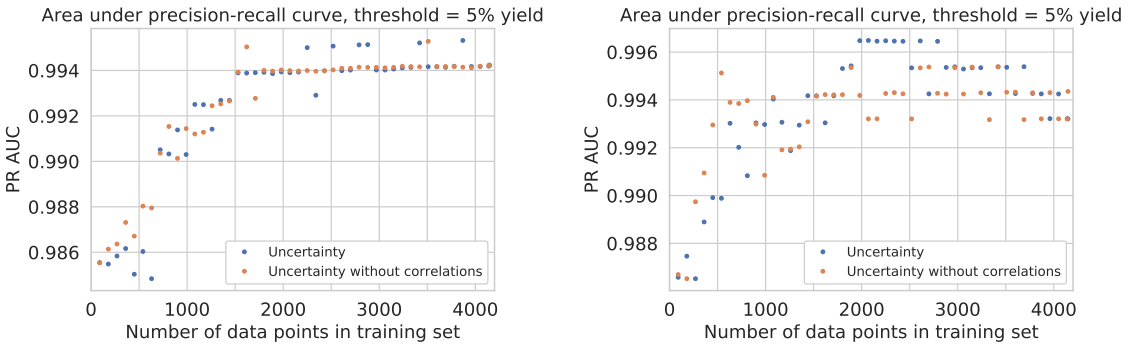
(a) Fold 1 **(b)** Fold 2

Figure 4.8: The difference $\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty}} - \mu_{\text{predictive variability}}^{\text{uncertainty without correlation}}$ of the mean predictive variabilities between uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations as a function of the number of points from the Pfizer data that have been added to the training set. Fold 1 and 2, of in total ten folds, of the cross validation are displayed. The red dotted line displays where the difference is equal to zero and the error bars show the 95% approximate confidence interval.



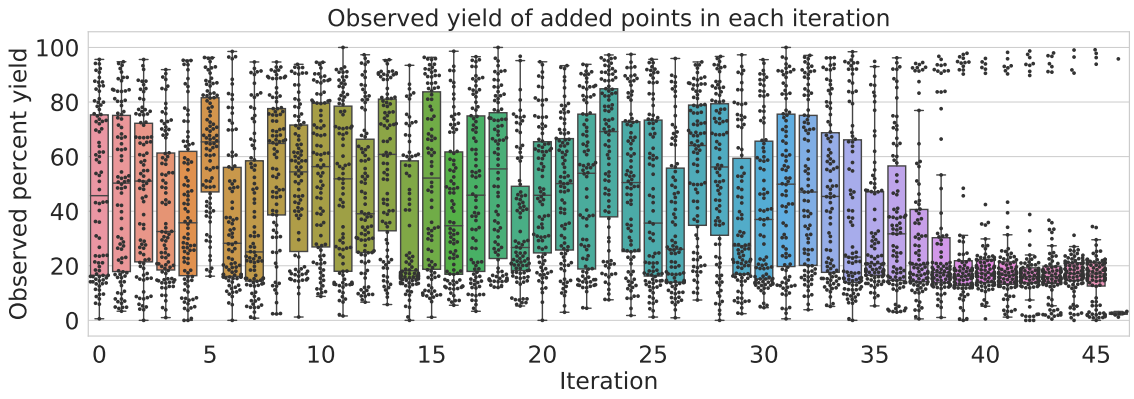
(a) RMSE **(b)** Predictive variability

Figure 4.9: The average (across all folds) differences $\Delta\mu_{\text{RMSE}}$ and $\Delta\mu_{\text{predictive variability}}$ of uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulation simulations as a function of the number of points from the Pfizer data that have been added to the training set. The red dotted line displays where the difference is equal to zero.

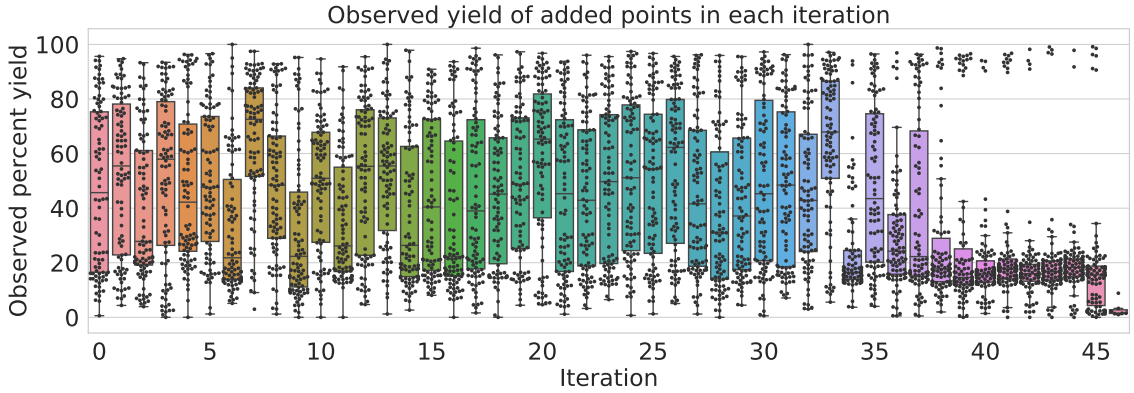


(a) Fold 1 **(b)** Fold 2

Figure 4.10: Area under precision-recall curve of uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations applied on the Pfizer data.



(a) Uncertainty sampling without correlation simulation



(b) Uncertainty sampling

Figure 4.11: Observed percent yield of points from the Pfizer data that have been added to the training set in each iteration when utilizing uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations. Only fold 1 is displayed but the other folds show similar behaviors. Iteration 0 is the initialization, which was done by random sampling.

4.1.3 Uncertainty Compared to Random

This section presents the results of the comparison between uncertainty sampling without correlation simulations and random sampling obtained on the Merck data. The corresponding results obtained on the Pfizer data are displayed in Appendix A.1.3.1. The results obtained on the Pfizer data are similar to the results obtained on the Merck data, which are presented in this section.

4.1.3.1 Merck Data

Figure 4.12 shows the difference

$$\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{random}}$$

of the mean RMSEs between uncertainty sampling without correlation simulations and random sampling obtained on fold 1 and 2 of the Merck data. The approximate 95% confidence intervals are shown as error bars. For fold 1, as seen in Figure 4.12a, random sampling displays a lower mean RMSE, compared to uncertainty sampling without correlation simulations until around iteration 8 (where the first iteration is the random initialization and is denoted as the 0th iteration or iteration 0), where 810 points have been added. From around iteration 17 and onward, where 1620 points have been added, uncertainty sampling without correlation simulations starts to display a lower mean RMSE. The result of fold 2 is displayed in Figure 4.12b. During the first six iterations (including the initialization), the strategy with the lowest mean RMSE varies between each iteration. From the 6th iteration and onward, uncertainty sampling without correlation simulations demonstrate a lower mean RMSE. Furthermore, Figure 4.14a shows the average difference over all ten folds. It illustrates similar differences in RMSE as was displayed for fold 1 in Figure 4.12a, where random sampling displays a lower mean RMSE in the beginning, but is also similar to fold 2 in Figure 4.12b since it shows a lower RMSE after the initial iterations. The minimum average difference is approximately -1.3. The results illustrates that random sampling performs better than uncertainty sampling without correlation simulations when the training set is small. As the size of the training set increases, uncertainty sampling without correlation simulations consistently illustrates better performance than random sampling.

Figure 4.13 shows the difference

$$\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty}} - \mu_{\text{predictive variability}}^{\text{random}}$$

of the mean predictive variability between uncertainty sampling without correlation simulations and random sampling as a function of the number of points of the Merck data that have been added to the training set. Only fold 1 and 2 are presented here but the other folds show similar behaviors. The error bars show the 95% approximate confidence interval. For fold 1, as seen in Figure 4.13a, uncertainty sampling without correlation simulations demonstrate a lower mean predictive variability compared to random sampling after the nine initial iterations (including the random initialization which is denoted as the 0th iteration). Then the difference mostly displays values between -6 and 0. Fold 2, see Figure 4.13a, shows a lower mean predictive variability

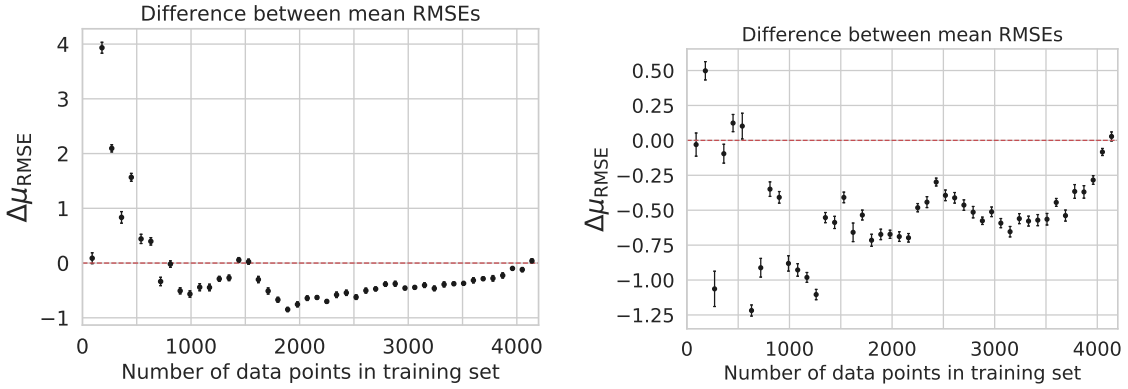
of uncertainty sampling without correlation simulations after the 13th iteration. Figure 4.14b shows the average difference over all folds. It demonstrates a difference similar to both fold 1 and 2 and the minimum average difference is approximately -3 which is obtained when 1530 points have been added to the training set. That $\Delta\mu_{\text{predictive variability}}$ eventually becomes negative indicates that, as the training set increases, uncertainty sampling without correlation simulations is more consistent in its predictions than random sampling.

Figure 4.15a shows the mean RMSE obtained on fold 1 of the Merck data as a function of the number of data points in the training set. The mean RMSE decreases with an increasing number of points in the data and reaches a value around 5.7 in the last iteration. Similar behaviors were also observed in the other folds. Moreover, Figure 4.15b shows the mean predictive variability as a function of the number of points in the training set. The mean predictive variability decreases with an increasing number of iterations and ends with a variance around 13. The other folds demonstrate similar behaviors.

Figure 4.16 shows the area under the precision-recall curve (PR AUC) score as function of the number of data points of the Merck data that have been added to the training set. Only the results of fold 1 and 2 are presented here but the other folds show similar behaviors. For fold 1, as seen in Figure 4.16a, the sampling strategy with the highest PR AUC score varies between each iteration until 7th iteration (where the first iteration is the random initialization and is denoted as the 0th iteration). From the 7th iteration and onward, the PR AUC score of random sampling stays lower than the PR AUC score of uncertainty sampling without correlation simulations. The PR AUC scores reaches values between 0.96 and 0.97 in the end, where the difference between the methods should be due to the stochastic behavior in the model since they in the end have the same training set. For fold 2, as demonstrated in Figure 4.16b, random sampling displays a higher PR AUC score in the beginning but uncertainty sampling without correlation simulations shows a higher PR AUC score after the 8th iteration. The PR AUC scores for both strategies reach values of about 0.980.

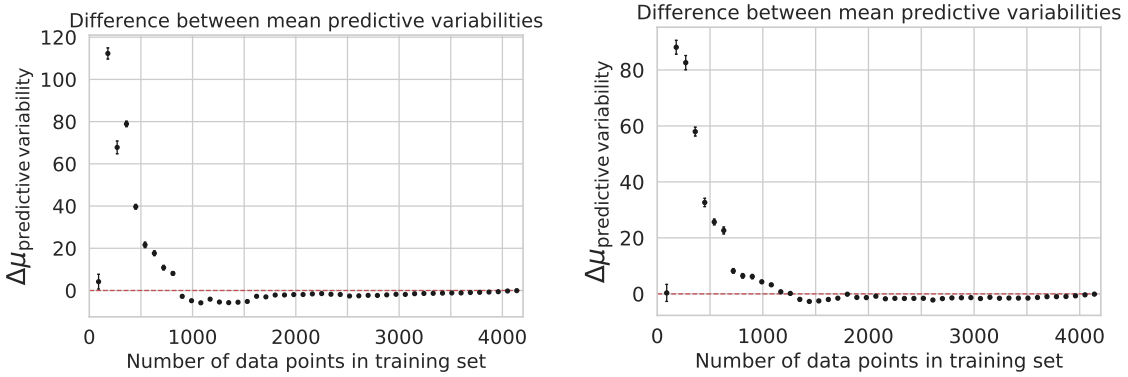
Figure 4.17 shows the observed yield that was added in each iteration of uncertainty sampling without correlation simulations and random sampling. This is shown for fold 1 but the other folds show similar behaviors. As demonstrated in previous results and again seen in Figure 4.17a, uncertainty sampling without correlation simulations adds a majority of points with yield lower than 40% in the last nine iterations and a majority of points with yields lower than 10% in the last five iterations. Moreover, as seen in Figure 4.17b, random sampling instead add points with different yields in each iteration, which is to be expected since it samples randomly from the available points. Figure 4.18 show the two-sided p-value from a Mann-Whitney U test between uncertainty sampling without correlation simulations and random sampling. The p-value is shown for all number of data points of fold 1 and 2 after the initialization, i.e from 91 points in the training set and onward. For both folds, the p-value is high in the beginning and decreases when the number of points is increased, except for some spikes in the p-value. After around 250 points have been added to the training set, both folds obtain a p-value lower than 0.05.

The p-value then stay below 0.05 until it starts to increase when the last number of data points is added to the training set, which is to be expected since then the two data sets contain the same points. This indicates that the sets generated by uncertainty sampling without correlation simulations and random sampling are significantly different, except in the beginning and the end.



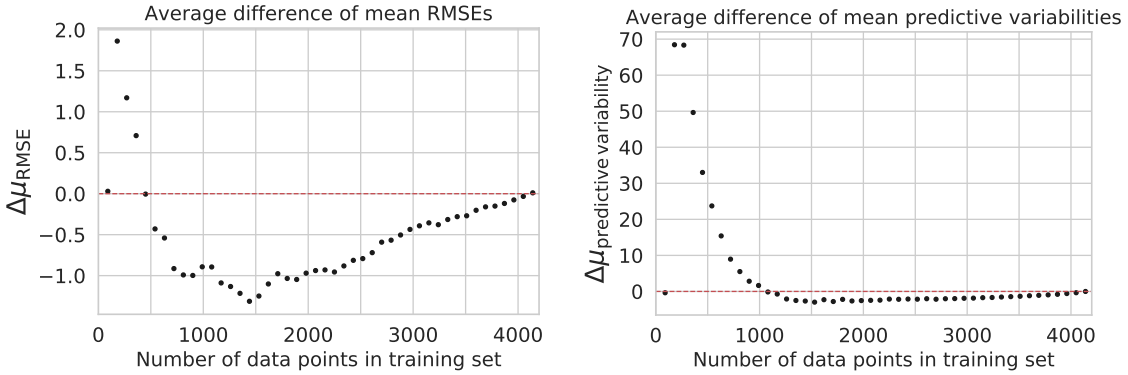
(a) Fold 1 **(b)** Fold 2

Figure 4.12: The difference $\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{random}}$ of the mean RMSEs between uncertainty without correlation simulations and random sampling as a function of the number of points of the Merck data that have been added to the training set. Fold 1 and 2, of in total ten folds, of the cross validation is displayed. The red dotted line displays where the y-axis is equal to zero and the error bars display the approximate 95% confidence intervals.



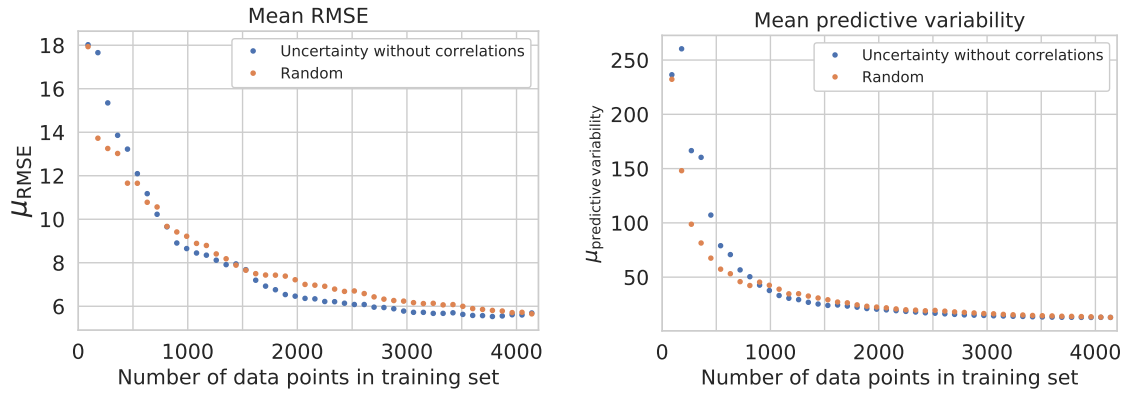
(a) Fold 1 **(b)** Fold 2

Figure 4.13: The difference $\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty}} - \mu_{\text{predictive variability}}^{\text{random}}$ of the mean predictive variabilities of uncertainty without correlation simulations and random sampling as a function of the number of points from the Merck data that have been added to the training set. Folds 1 and 2, of in total ten folds, of the cross validation is displayed. The red dotted line displays where the y-axis is equal to zero and the error bars display the approximate 95% confidence intervals.



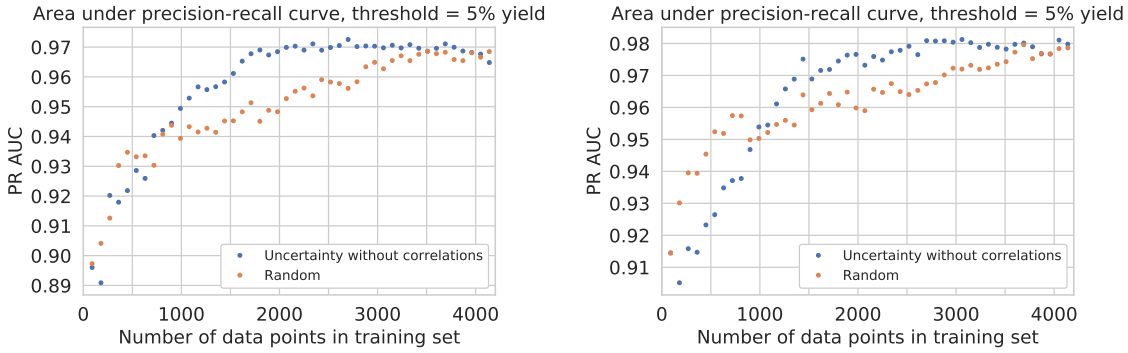
(a) RMSE **(b)** Predictive variability

Figure 4.14: The average differences $\Delta\mu_{\text{predictive variability}}$ and $\Delta\mu_{\text{RMSE}}$ between uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Merck data that have been added to the training set. The red dotted line displays where the difference is equal to zero.



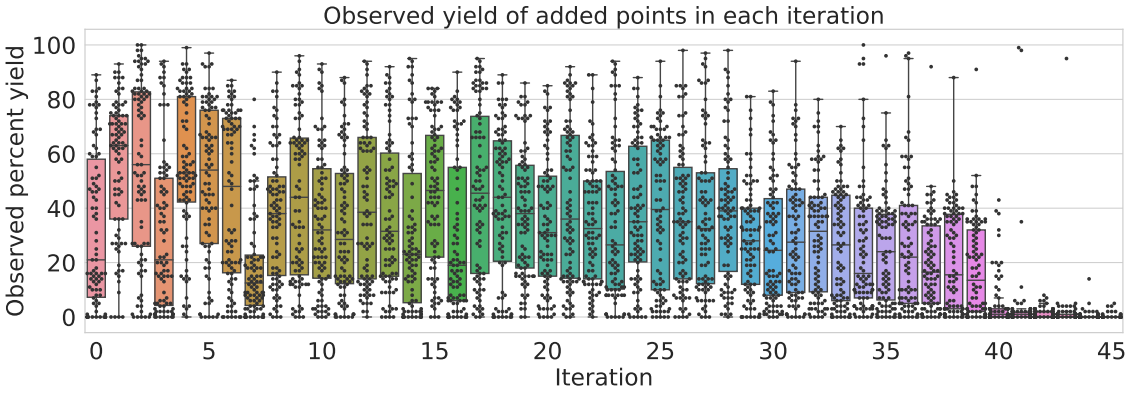
(a) RMSE **(b)** Variance

Figure 4.15: Mean RMSE and mean predictive variability of uncertainty sampling without correlation simulations and random sampling over all predictions as a function of the number of points from the Merck data that have been added to the training set. Results for only fold 1 are shown but the other folds show similar convergences.

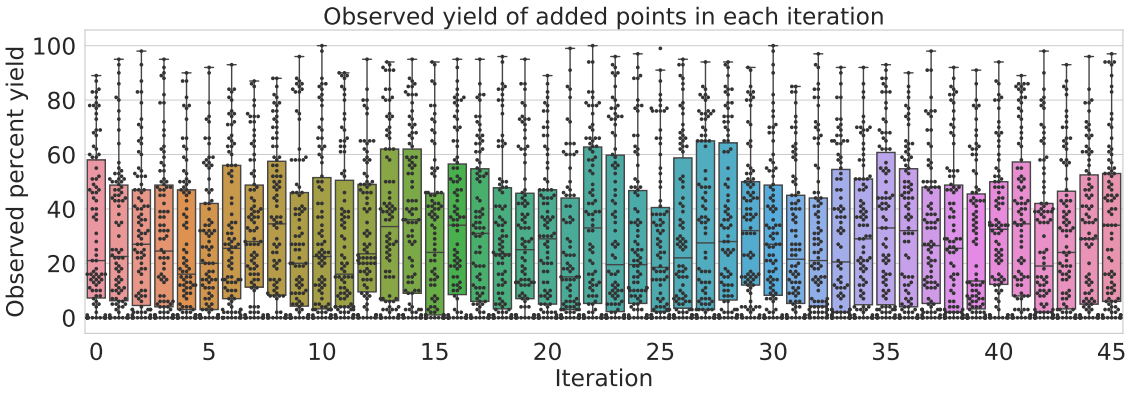


(a) Fold 1 **(b)** Fold 2

Figure 4.16: Area under precision-recall curve of uncertainty sampling without correlation simulations and random sampling applied on the Merck data.

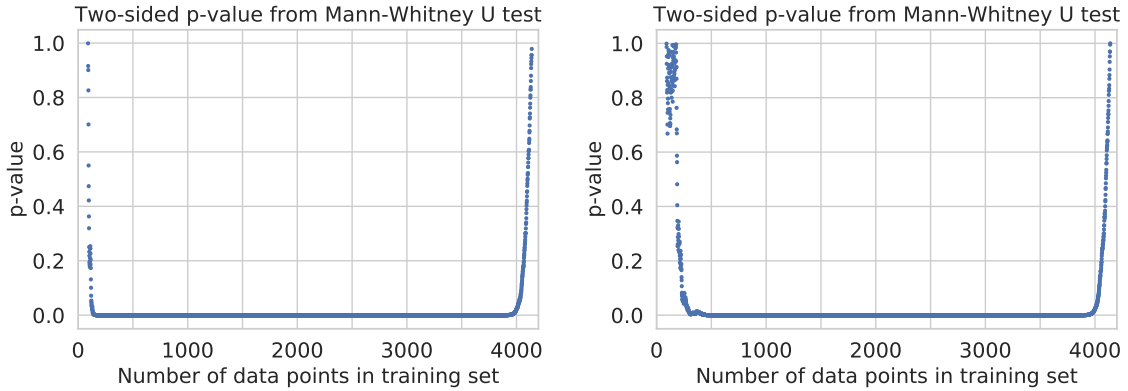


(a) Uncertainty sampling without correlation simulation



(b) Random sampling

Figure 4.17: Observed percent yield of points from the Merck data that have been added to the training set in each iteration when utilizing uncertainty sampling without correlation simulations and random sampling. Only fold 1 is displayed but the other folds show similar behaviors.



(a) Fold 1 **(b)** Fold 2

Figure 4.18: The p-values of a Mann-Whitney U test of all subsets of the training set with points from the Merck data when utilizing uncertainty sampling without correlation simulations and random sampling. p-values are not plotted for the subsets of the initialization. Only fold 1 and 2 is displayed but the other folds show similar behaviors. For a sufficiently low p-value (depending on the desired significance level), the null hypothesis that the sets are from the same distribution can be rejected.

4.1.4 Uncertainty Compared to Kennard-Stone Algorithm

This section presents the results of the comparison between uncertainty sampling without correlation simulations and the Kennard-Stone algorithm when both strategies utilized the Kennard-Stone algorithm as initialization. The comparison was only done for the Merck data since the Pfizer data have shown the same trends in the other comparisons.

4.1.4.1 Merck Data

Figure 4.19 shows the difference

$$\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{Kennard-Stone}}$$

of the mean RMSEs between uncertainty sampling without correlation simulations and the Kennard-Stone algorithm applied on fold 1 and 2 of the Merck data. For both fold 1 and 2, uncertainty sampling without correlation simulations seems to yield a significantly lower RMSE after a few iterations. The same behavior is also illustrated in Figure 4.21a which shows the average difference of the average, across all folds, differences of the mean RMSEs. The minimum average difference across all folds is around -1.1. This illustrates that uncertainty sampling without correlation simulations performs better with respect to RMSE than the Kennard-Stone algorithm.

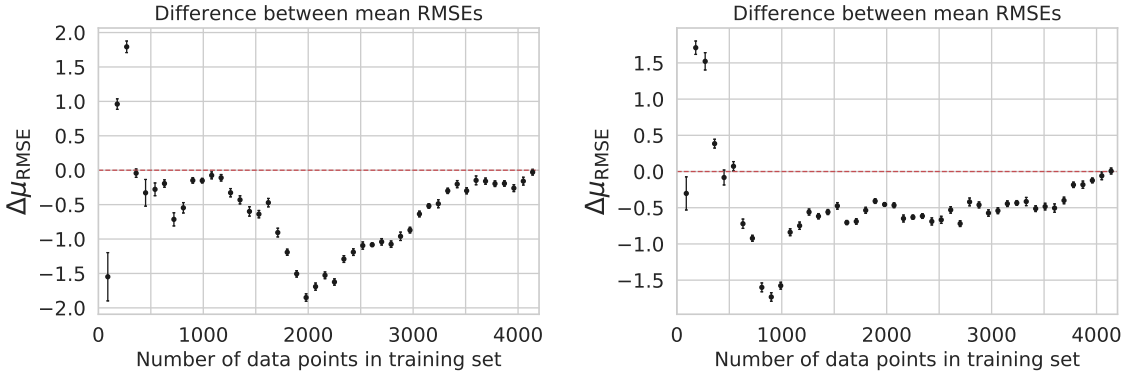
Figure 4.20 displays the difference

$$\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty}} - \mu_{\text{predictive variability}}^{\text{Kennard-Stone}}$$

of the mean predictive variabilities between uncertainty sampling without correlation simulations and the Kennard-Stone algorithm applied on fold 1 and 2 of the Merck data. For fold 1, as seen in Figure 4.20a, the strategy with the lowest predictive variability varies between each iteration until the 15th iteration (where the first iteration is the initialization using the Kennard-Stone algorithm and is denoted as the 0th iteration). Form the 15th iteration and onward, uncertainty sampling without correlation simulations shows a consistently lower mean predictive variability compared to the Kennard-Stone algorithm; but the difference is at some iterations (after the 15th iteration) close to zero and the minimum difference is around -3. For fold 2, as seen in Figure 4.20b, the Kennard-Stone algorithm seems to yield a lower mean predictive variability in the beginning and then the strategy with the lowest mean predictive variability varies between each iteration until the 20th iteration. From the 20th iteration and onward, uncertainty sampling without correlation simulations seems to yield a consistently lower variability but the minimum difference is around -3. Figure 4.21b shows the average, across all folds, difference of the mean predictive variability. The minimum average difference when excluding the 0th iteration (the initialization) is around -2.7. Similar to separate inspections of fold 1 and 2, the average difference displays a lower variability of the Kennard-Stone algorithm in the beginning. On the other hand, uncertainty sampling without correlation simulations shows a consistently lower variability, compared to the Kennard-Stone algorithm, after a sufficient number of iterations. The results indicate that as the

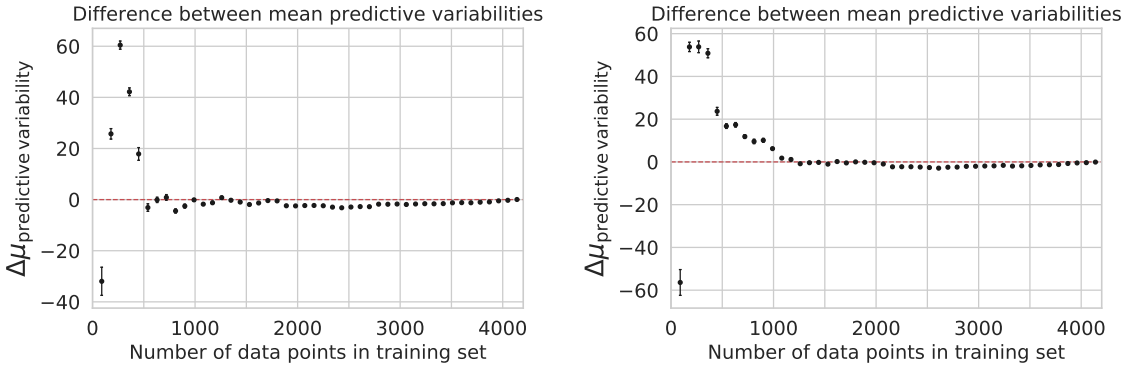
training set increases, uncertainty sampling without correlation simulations is more consistent in its predictions than the Kennard-Stone algorithm.

Figures 4.22a and 4.22b display the area under the precision-recall curve (PR AUC) of uncertainty sampling without correlation simulations and the Kennard-Stone algorithm utilized on fold 1 and 2, respectively. After a sufficient number of iterations, uncertainty sampling without correlation simulations demonstrates a higher PR AUC score. Figures 4.23a and 4.23b display the observed yield that was added in each iteration when uncertainty sampling without correlation simulations and the Kennard-Stone algorithm, respectively, were applied on fold 1. The Kennard-Stone algorithm shows a behavior similar to random sampling (see Figure 4.17b) where different yields are added uniformly over the iterations. As previously observed, uncertainty sampling without correlation simulations adds a greater number of points with high yield in the beginning and a greater number of points with low yield in the end compared to the Kennard-Stone algorithm (and random sampling).



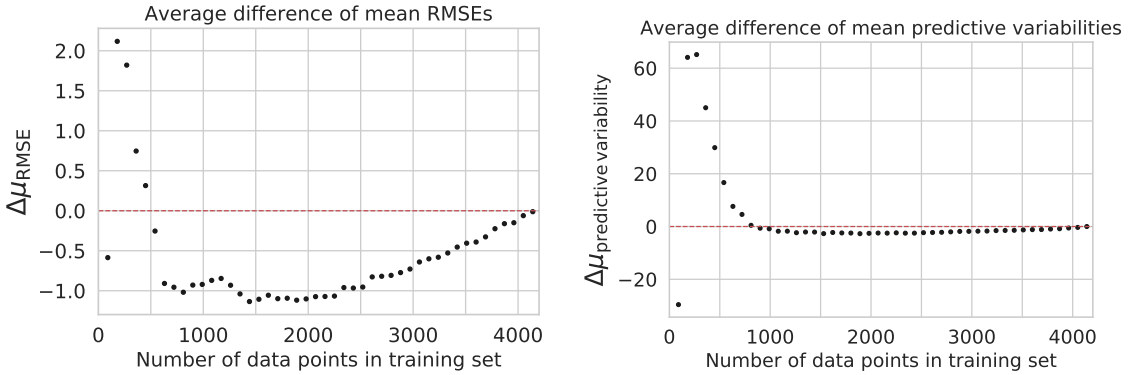
(a) Fold 1 **(b)** Fold 2

Figure 4.19: The difference $\Delta\mu_{\mathrm{RMSE}} = \mu_{\mathrm{RMSE}}^{\mathrm{uncertainty}} - \mu_{\mathrm{RMSE}}^{\mathrm{Kennard-Stone}}$ of the mean RMSEs of uncertainty sampling without correlation simulations and the Kennard-Stone algorithm as a function of the number of points from the Merck data that have been added to the training set.



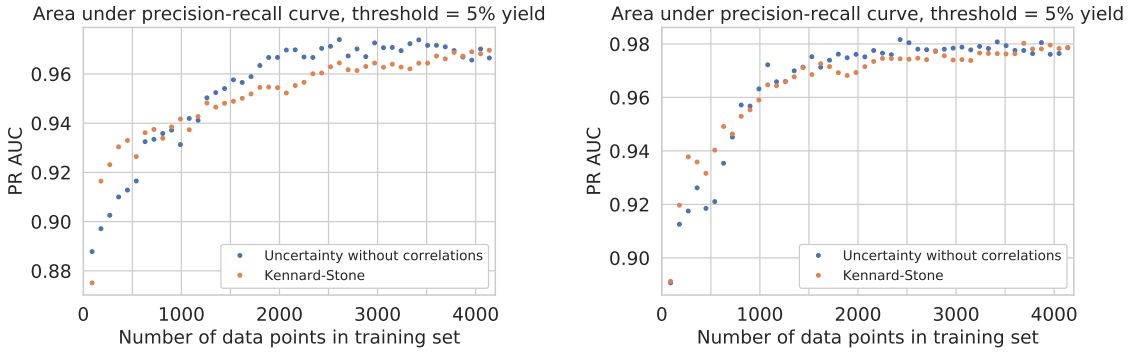
(a) Fold 1 **(b)** Fold 2

Figure 4.20: The difference $\Delta\mu_{\text{predictive variability}} = \mu^{\text{uncertainty}}_{\text{predictive variability}} - \mu^{\text{Kennard-Stone}}_{\text{predictive variability}}$ of the mean predictive variabilities between uncertainty sampling without correlation simulation simulations and the Kennard-Stone algorithm as a function of the number of points from the Merck data that have been added to the training set.



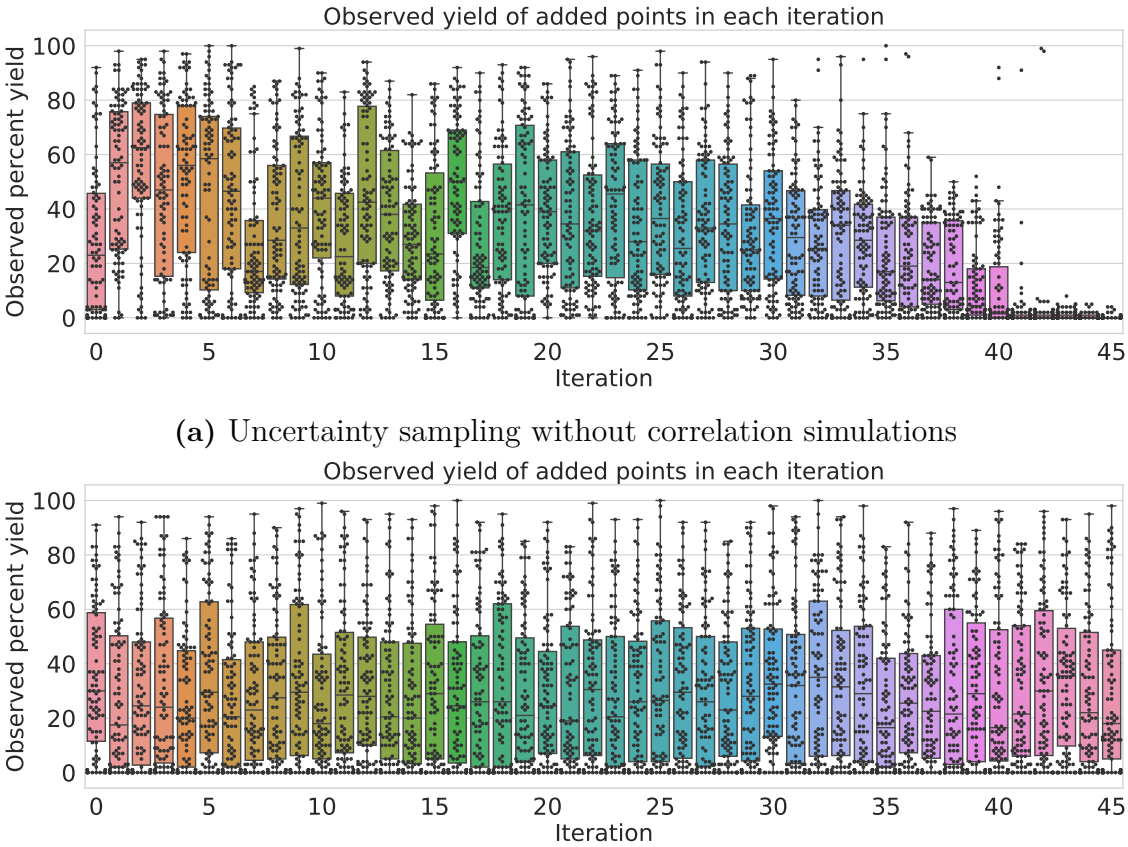
(a) RMSE **(b)** Predictive variability

Figure 4.21: The average differences $\Delta\mu_{\text{predictive variability}}$ and $\Delta\mu_{\text{RMSE}}$ of uncertainty sampling without correlation simulations and the Kennard-Stone algorithm as a function of the number of points from the Merck data that have been added to the training set.



(a) Fold 1 **(b)** Fold 2

Figure 4.22: Area under precision-recall curve of uncertainty sampling without correlation simulations and the Kennard-Stone algorithm obtained on the Merck data.



(a) Uncertainty sampling without correlation simulations

(b) Kennard-Stone algorithm

Figure 4.23: Observed percent yield of points of the Merck data that have been added to the training set in each iteration when utilizing uncertainty sampling without correlation simulations and the Kennard-Stone algorithm. Only fold 1 is displayed but the other folds show similar behaviors. Note that the initial sets are different since the ties as are resolved by randomly selecting points with the same minimum distance.

4.1.5 Uncertainty Compared to Random When Utilizing Fingerprints

This section displays the results of the comparison between uncertainty sampling without correlation simulations and random sampling when fingerprints of the reaction components are used as side information for Macau. The results was obtained on the Merck data using 2048-bits ECFP6 fingerprints with counts for the reaction components.

4.1.5.1 Merck Data

Figure 4.24 illustrates the difference

$$\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{random}}$$

of mean RMSEs between uncertainty sampling without correlation simulations and random sampling as function of the number of points from the fold 1 and 2 of the Merck that have been added to the training set. For fold 1, as seen in Figure 4.24a, uncertainty sampling without correlation simulations shows a significantly lower mean RMSE after the 4th iteration (where the first iteration is the random initialization and is denoted as the 0th iteration). The same behavior is seen for fold 2, as seen in Figure 4.24b, after the eight iteration. For both folds, the minimum difference takes on a value around -1.2. Figure 4.26a shows the average difference across all folds except fold 6 and 10. The average difference shows a lower RMSE after the 4th iteration which is a similar behavior to what was observed for fold 1 and 2. However, Figures 4.24a, 4.24b and 4.26a show a lower mean RMSE in the beginning, before the mean RMSE is significantly lower for uncertainty sampling without correlation simulations. Moreover, Figure 4.27a shows the mean RMSE of uncertainty sampling without correlation simulations and random sampling obtained from fold 1 of the Merck data. The difference in 4.24a is visible there as well. The mean RMSE obtained on fold 1 seems to reach a value around 5.6 at the last iteration. Finally, note that these results illustrate that uncertainty sampling without correlation simulations performs better than random sampling when the training set increases.

Figure 4.25 shows the difference

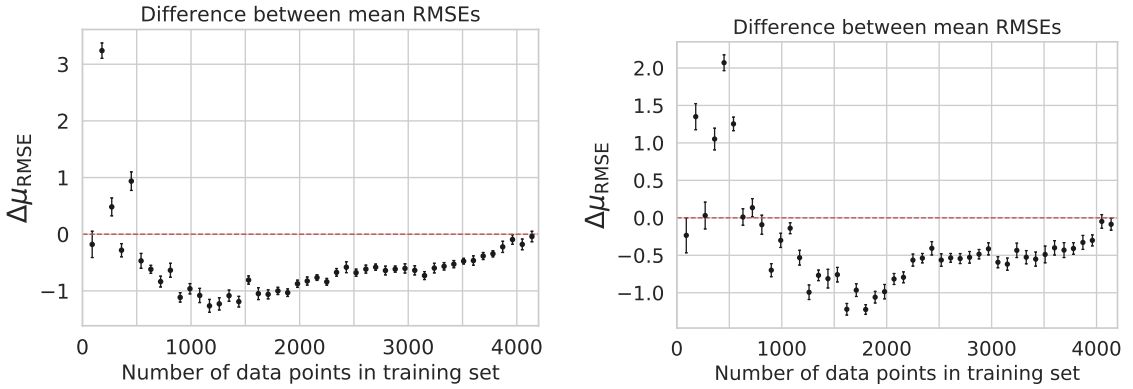
$$\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty}} - \mu_{\text{predictive variability}}^{\text{random}}$$

of the mean predictive variabilities of uncertainty without correlation simulations and random sampling as a function of the number of points from fold 1 and 2 of the Merck data that have been added to the training set. For fold 1, as seen in Figure 4.25a, uncertainty sampling without correlation simulations shows a significantly lower predictive variability after the 10th iteration; while fold 2, as seen in Figure 4.25b, show the same behavior after 14th iteration. The average difference across all folds is shown in Figure 4.26b. The average difference shows a lower mean RMSE for uncertainty sampling without correlation simulations after the 11th iteration. That $\Delta\mu_{\text{predictive variability}}$ eventually becomes negative illustrates that uncertainty sampling without correlation simulations is more consistent in its predictions than

random sampling. Moreover, Figure 4.27b presents the mean predictive variability of uncertainty sampling without correlation simulations and random sampling obtained on fold 1 of the Merck data. The figure displays mean predictive variabilities around 13 at the last iteration.

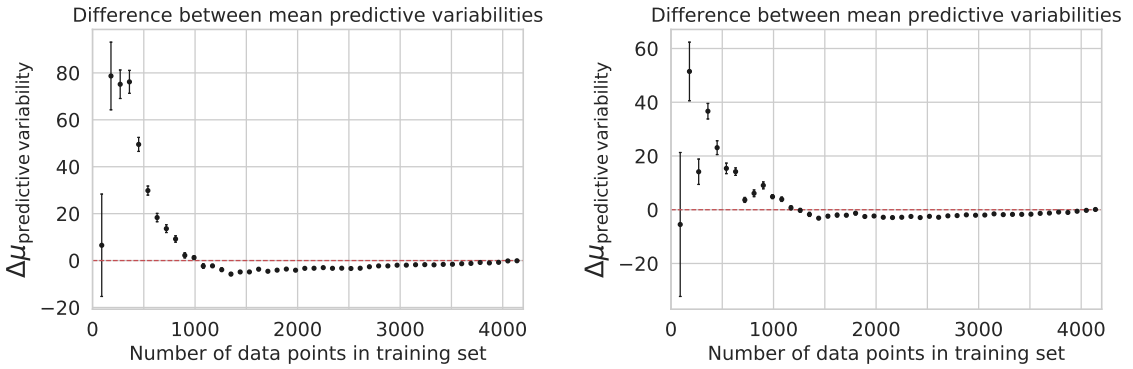
Figure 4.28 shows the area under the precision-recall curve (PR AUC) of uncertainty sampling without correlation simulations and random sampling obtained on fold 1 and 2 of the Merck data. For both folds, uncertainty sampling without correlation simulations shows a higher PR AUC score after the 10th iteration and onward (where the first iteration is the random initialization and is denoted as the 0th iteration). The PR AUC scores of fold 1 reach values between 0.973 and 0.977 in the end; while the PR AUC score of fold 2 reaches a value of 0.983 for both strategies in the end.

Figure 4.29 shows the observed yields of points from fold 1 that were added in each iteration of uncertainty sampling without correlation simulations and random sampling. For uncertainty sampling without correlation simulations, 4.29a, mostly points with a yield lower than 20% are added to the training set in the last iterations. Hence, it seems to favor the selection of points with higher yields. Random sampling does not favor any yield as expected. Figure 4.30 displays the two-sided p-value from Mann-Whitney U test between uncertainty sampling without correlation simulations and random sampling obtained on fold 1 and 2. The p-values is shown for all number of data points after the initialization, i.e from 91 points in the training set and onward. Both folds show a p-value lower than 0.05 after around 500 points have been added to the training set which means that the two training sets are significantly different. As expected, the value starts to increase in the end since then the two training sets contain the same data.



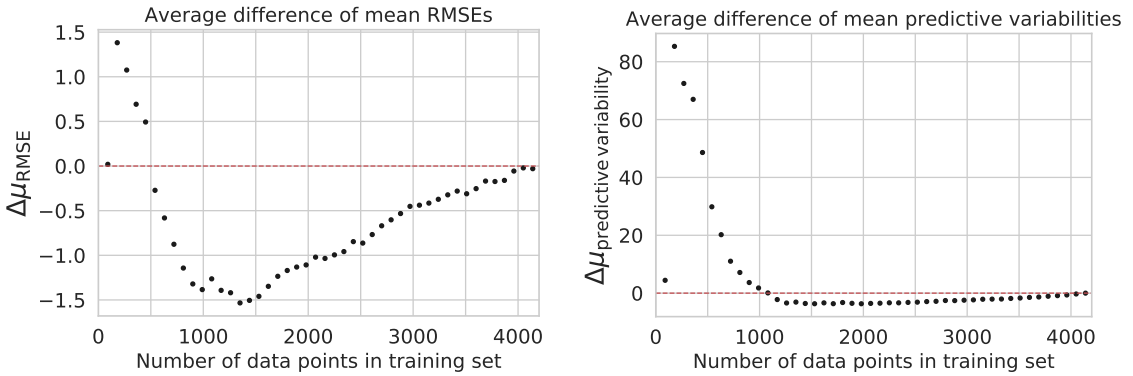
(a) Fold 1 **(b)** Fold 2

Figure 4.24: The difference $\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{random}}$ of the mean RMSEs between uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Merck data that have been added to the training set. Fingerprints were used as side information to Macua. Fold 1 and 2, of in total ten folds, of the cross validation are displayed. The red dotted line displays where the difference is equal to zero and the error bars display the approximate 95% confidence intervals.



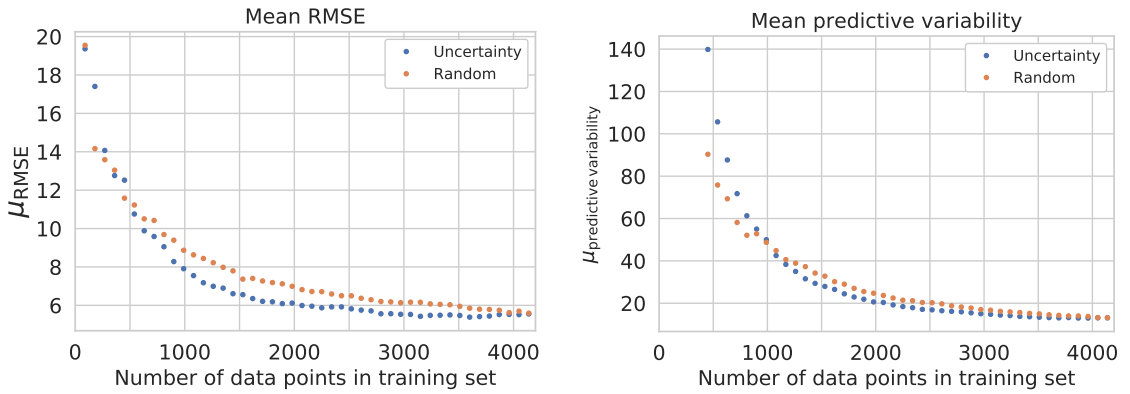
(a) Fold 1 **(b)** Fold 2

Figure 4.25: The difference $\Delta\mu_{\text{predictive variability}} = \mu^{\text{uncertainty}}_{\text{predictive variability}} - \mu^{\text{random}}_{\text{predictive variability}}$ of the mean predictive variabilities of uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Merck data that have been added to the training set. Fingerprints were used as side information to Macua. Fold 1 and 2, of in total ten folds, of the cross validation are displayed. The red dotted line displays where the difference is equal to zero and the error bars display the approximate 95% confidence intervals.



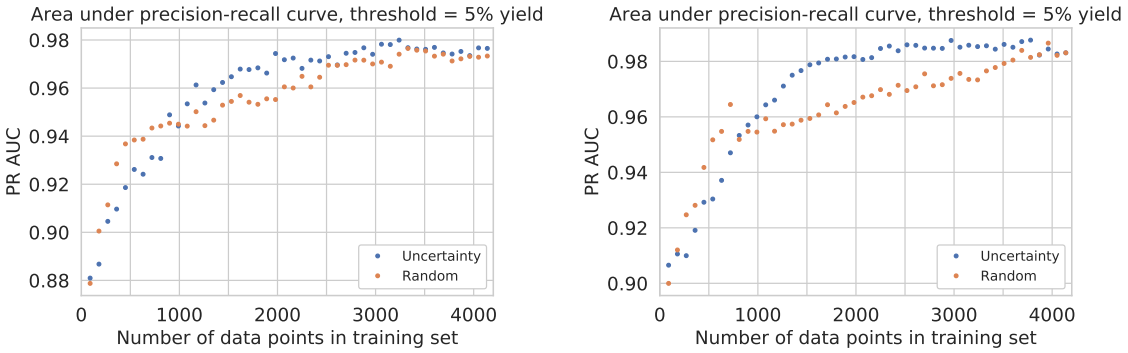
(a) RMSE **(b)** Predictive variability

Figure 4.26: The average differences $\Delta\mu_{\text{RMSE}}$ and $\Delta\mu_{\text{predictive variability}}$ between uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Merck data that have been added to the training set. Fingerprints were used as side information to Macau and the average is across all folds. The red dotted line displays where the difference is equal to zero.



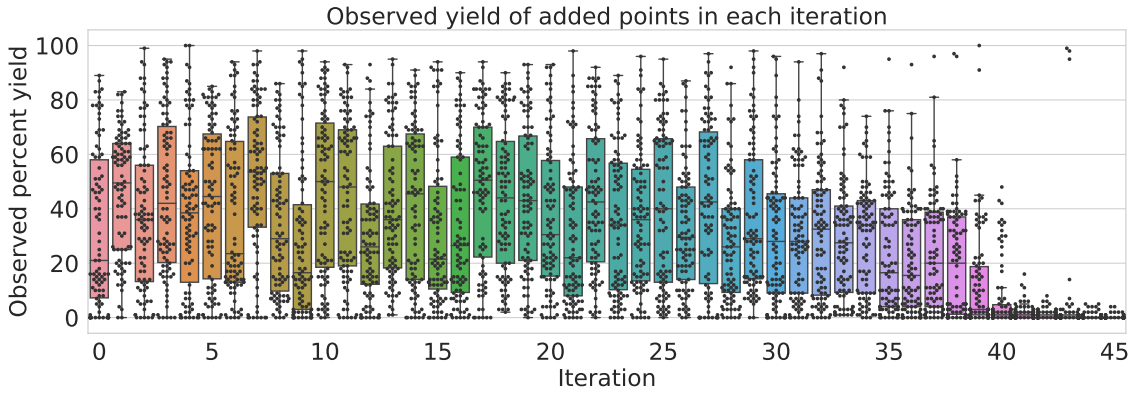
(a) RMSE **(b)** Predictive variability

Figure 4.27: The mean RMSE and mean predictive variability of uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Merck data that have been added to the training set. Fingerprints were used as side information to Macau. Only the results for fold 1 are shown, but the other folds show similar convergences.

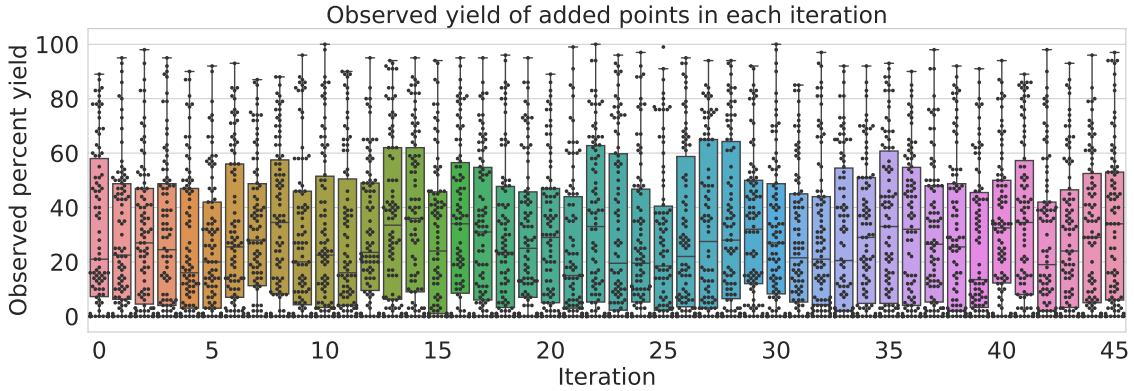


(a) Fold 1 **(b)** Fold 2

Figure 4.28: Area under precision-recall curve of uncertainty sampling without correlation simulations and random sampling obtained on the Merck data. Fingerprints were used as side information to Macau. Only results for fold 1 and 2 are displayed.

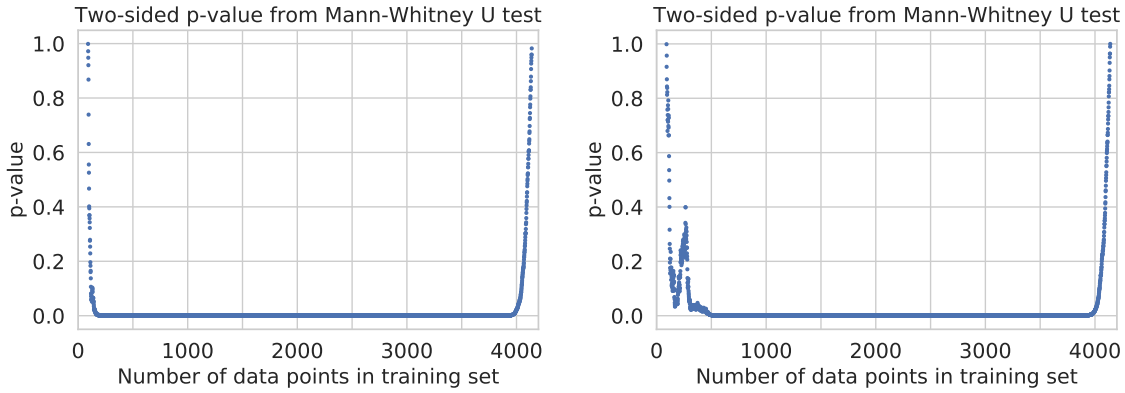


(a) Uncertainty sampling without correlation simulations



(b) Random sampling

Figure 4.29: Observed percent yield of points from the Merck data that have been added to the training set in each iteration when utilizing uncertainty sampling without correlation simulations and random sampling. Fingerprints were used as side information to Macau. Only fold 1 is displayed but the other folds show similar behaviors.



(a) Fold 1 **(b)** Fold 2

Figure 4.30: p-values of a Mann-Whitney U test of all subsets of the training set with points from the Merck data when utilizing uncertainty sampling without correlation simulations and random sampling. Fingerprints were used as side information to Macau. p-values are not plotted for the subsets of the initialization. Only fold 1 and 2 are displayed but the other folds show similar behaviors.

4.2 AZ ELN Performance

This section presents some of the predictive results obtained by Bayes by Backprop, see Section 3.2.2.1, and MC dropout, see Section 3.2.2.2, on the AZ ELN data. Recall that the networks are used for a variant of retrosynthesis prediction; in particular, they try to predict which ligand that was used in a reaction. Bayes by Backprop illustrates inferior accuracy and predicted diversity but better likelihood loss than MC dropout.

4.2.1 Bayes by Backprop

This section presents the predictive performance on the first CV fold for the Bayes by Backprop approach. The results for the second CV fold are presented in Appendix A.2.1, and the remaining folds illustrate similar trends.

Figure 4.31 illustrates the likelihood loss

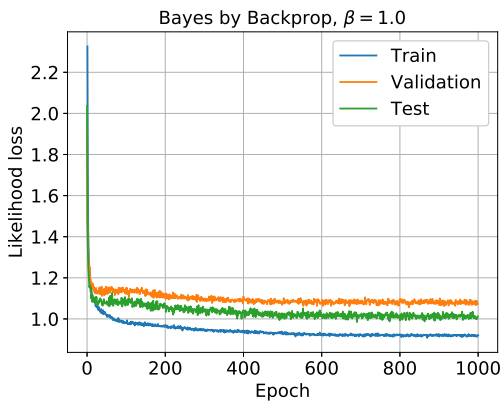
$$\frac{1}{T}\sum_{t=1}^{T}\log\left(\pi\left(\mathcal{D}|\boldsymbol{w}^{(t)}\right)\right),$$

and the complexity loss

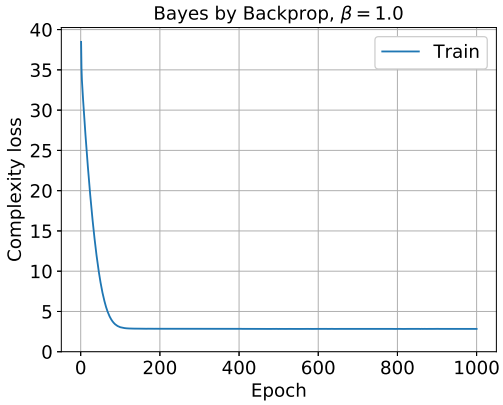
$$\frac{\beta}{T}\sum_{t=1}^{T}\log\left(\frac{q(\boldsymbol{w}^{(t)}|\theta)}{\pi(\boldsymbol{w}^{(t)})}\right)$$

averaged over the number of observations in each set, for $\beta \in \{1.0, 0.5, 0.1\}$. As previously noted, the complexity loss is data independent, which means that it is constant over the three data splits, i.e., the training, validation and test set. Therefore, only the complexity loss of the training data is presented in Figure 4.31. Furthermore, Figure 4.31 highlights that the main contribution to the total loss, which is the sum of the likelihood and complexity loss, is the complexity loss for all values of β. The fact that the loss is based upon two terms is a distinct difference to ordinary networks, where the classification loss usually is based solely upon the negative log likelihood, and maybe some weight decay. While the negative log likelihood loss also is present in Bayes by Backprop (it is the likelihood loss), the model is also regularized to be similar to the prior distribution by the complexity loss. This makes the model less complex and more resilient to overfitting. That the training, validation and test loss in Figure 4.31a are similar illustrates that the model does not overfit the data. While it is beneficial that the model is resilient to overfitting, it also induces a problem that the choice of the prior can be influential.

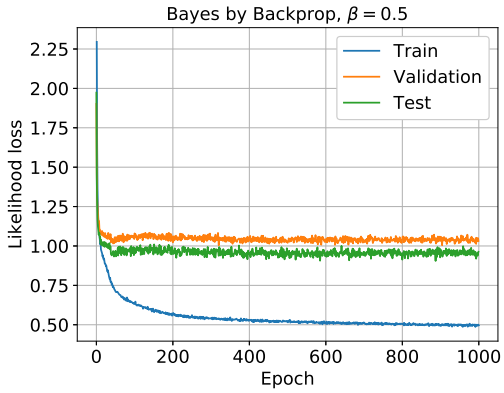
The results in Figure 4.31 also illustrate the effect of β. The model with $\beta = 1.0$ has the largest regularizing term, and as a consequence the model seems to be resilient to overfitting. This is clearly illustrated by the fact that the likelihood losses of the training, validation and test set are similar, as seen in Figure 4.31a. This indicates that the learnt features generalizes from the training to the validation and test set. For smaller values of β the model starts to overfit the data. This is most clear for $\beta = 0.1$ where the likelihood losses of the validation and test set increases after the initial epochs, as seen in Figure 4.31e. Furthermore, even though $\beta = 0.5$ illustrates some overfitting, as seen in Figure 4.31c, the test and validation loss is also lower



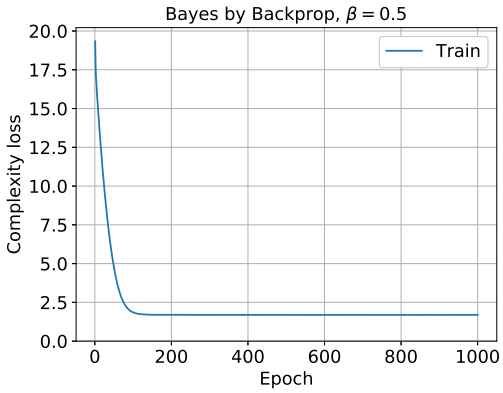
(a) Likelihood loss (Negative log likelihood)



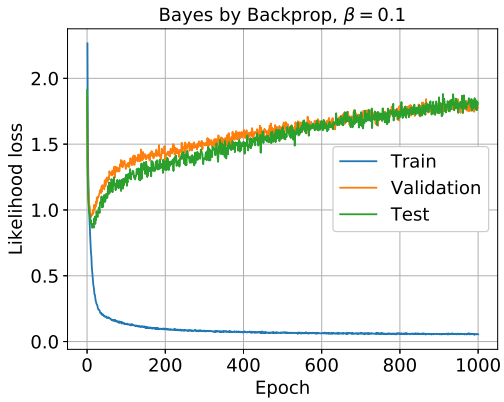
(b) Complexity loss



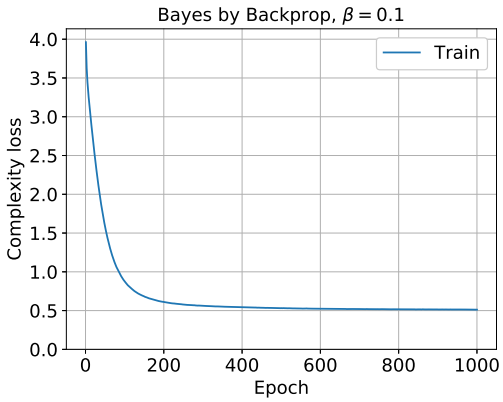
(c) Likelihood loss (Negative log likelihood)



(d) Complexity loss



(e) Likelihood loss (Negative log likelihood)



(f) Complexity loss

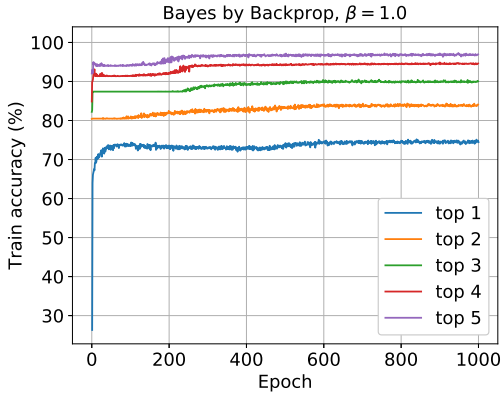
Figure 4.31: The likelihood and complexity loss on the first CV fold of the AZ ELN data. The loss is averaged over the number of data points in each set. Note that since the complexity loss does not depend on the data it is constant between the train, validation and test split. Furthermore, note that the training likelihood loss decreases with β.

than for $\beta = 1.0$ which indicates that the model better represents the data. Finally, note that the complexity loss is larger than the likelihood loss for all tested values of β.

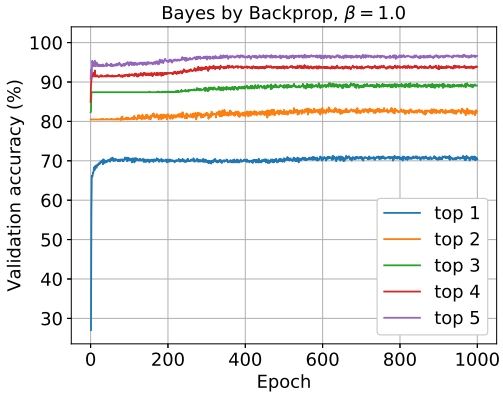
Figure 4.32 illustrates the top n accuracy (for $n \in \{1, \ldots, 5\}$) for the first CV fold of the AZ ELN data of the Bayes by Backprop approach. The top n accuracies are increasing in the initial epochs and then show no significant increase, but instead stay at the same level. Therefore, these results indicate that the model initially learns some features of the data, however, it struggles to learn more features that generalize to both the validation and test set. As mentioned above, the results in Figure 4.31a indicate that the model does not overfit the data, due to the similarity between the training, validation and test set. This is further verified by the fact that the top n accuracy also is similar among the sets. Furthermore, that $\beta = 0.1$ overfits the data is clearly visualized by the fact that the training accuracy tends to 100%, as seen in Figure 4.32g, while the test accuracy decreases, as seen in Figure 4.32i. Finally, note that the top 5 accuracy for the validation and test set is similar for all values of β, and that it is consistently around 98%.

The predictive diversity of Bayes by Backprop for $\beta \in \{1.0, 0.5, 0.1\}$ is visualized in Figure 4.33. It is worth noting that when the $\beta = 1.0$ model predicts ligand 12, after around 450 epochs, the accuracy of the model also increases for all three splits, as displayed in Figure 4.32. The first three ligands that the $\beta = 1.0$ model manages to find are the most frequent ligands and they constitute approximately 87% of the data. However, the model does not predict ligand 3, which is twice as frequent as ligand 12. Note that the $\beta = 1.0$ model only manages to predict a third of the possible ligands. The poor performance of the model might be explained by the severely imbalanced data, or by the large complexity loss.

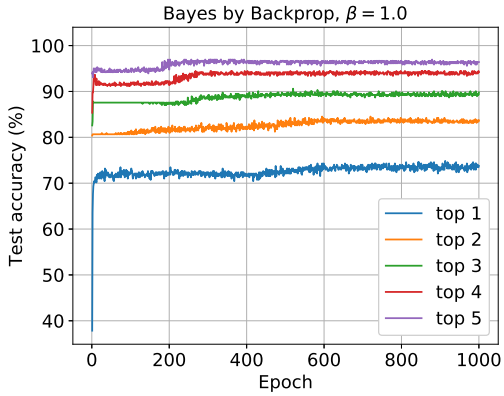
Figure 4.33 also illustrates that the model predicts more ligands for lower values of β. This is coherent with the results presented in Figure 4.31 as the (training) likelihood loss also decreased with β. That the likelihood loss decreases with β is to be expected since the model then is more encouraged to fit the data, thereby forcing the model to predict more ligands. A consequence of this is that the low predicted diversity for the $\beta = 1.0$ model probably originates in the large complexity loss. Note that an increasing number of predicted ligands do not necessarily correspond to an increase in the top 1 accuracy. This is illustrated by the fact that the $\beta = 0.5$ model has a higher test top 1 accuracy, see Figure 4.32f, than the $\beta = 0.1$ model, see Figure 4.32i, even though it only predicts half as many ligands. That $\beta = 0.1$ manages to predict most of the ligands, without increasing the validation and test accuracy, further implies that the model overfits the data; since this means that the model is confident in more ligands without actually increasing the performance.



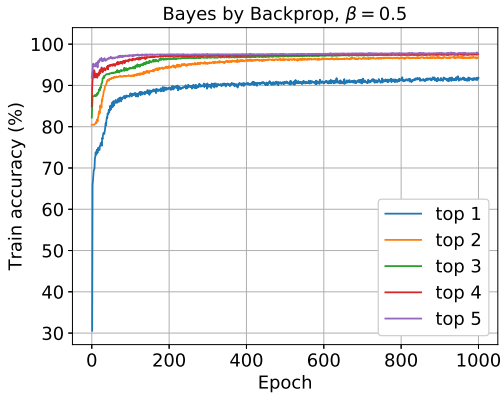
(a) Train



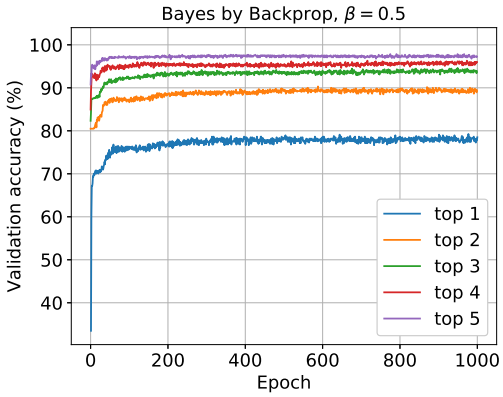
(b) Validation



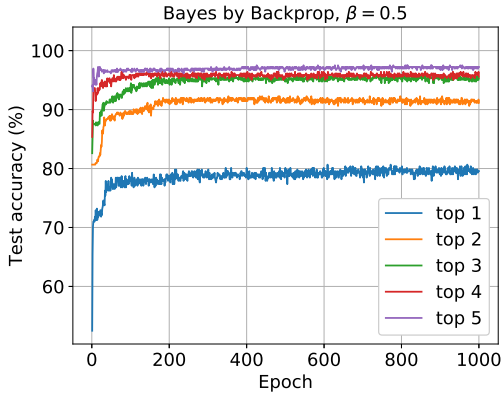
(c) Test



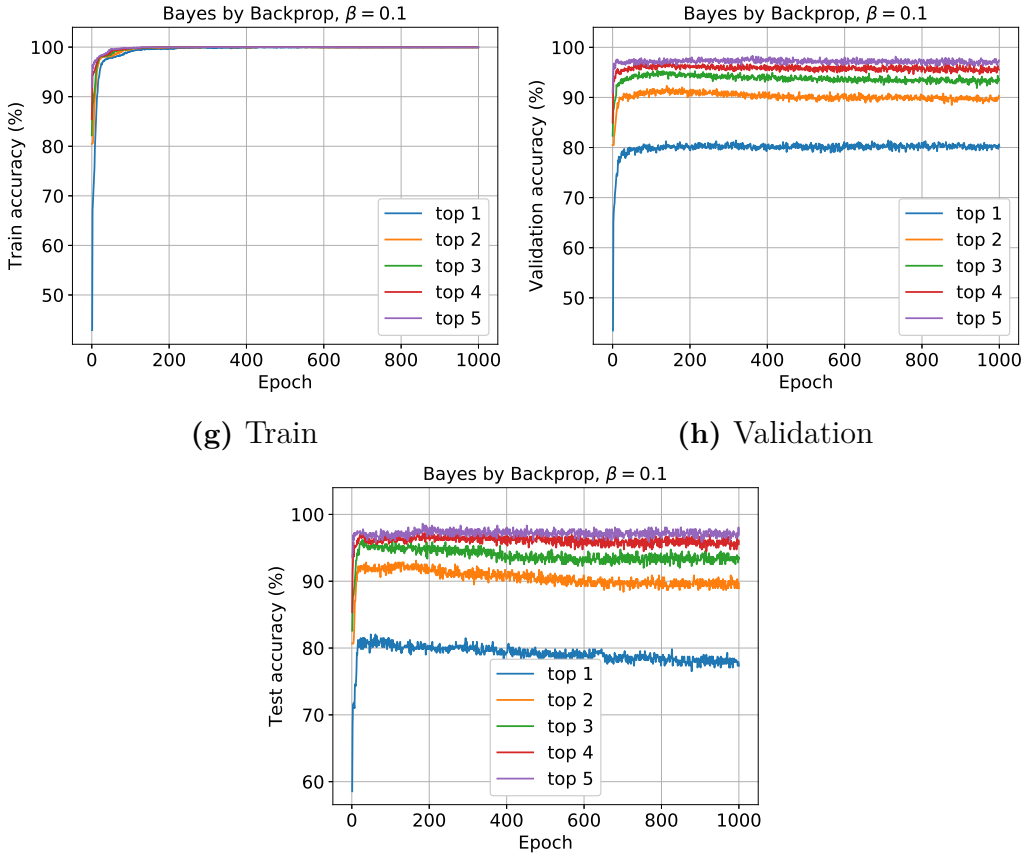
(d) Train



(e) Validation



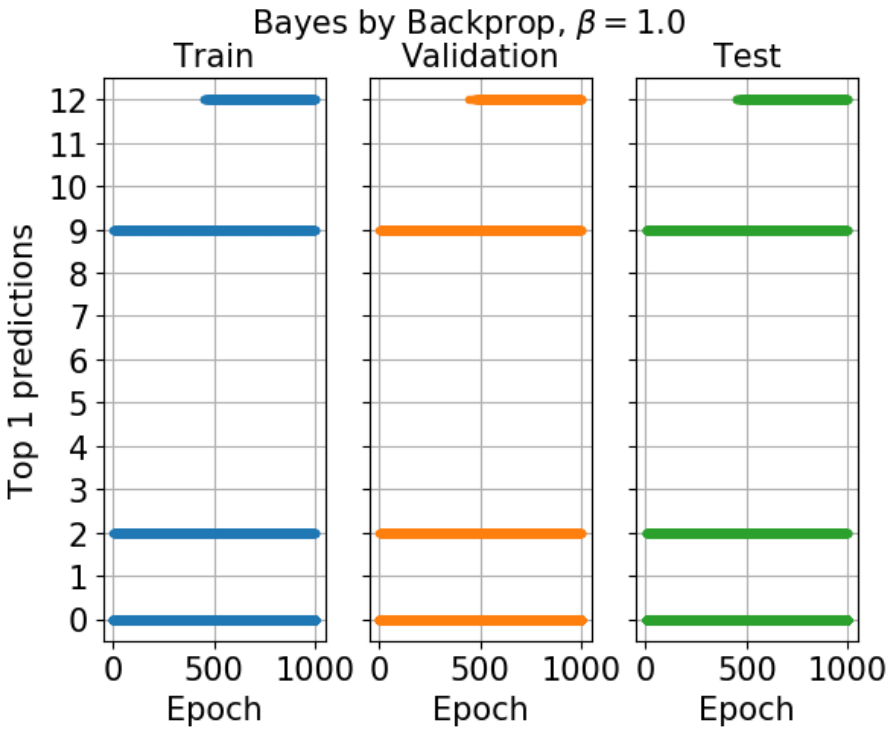
(f) Test



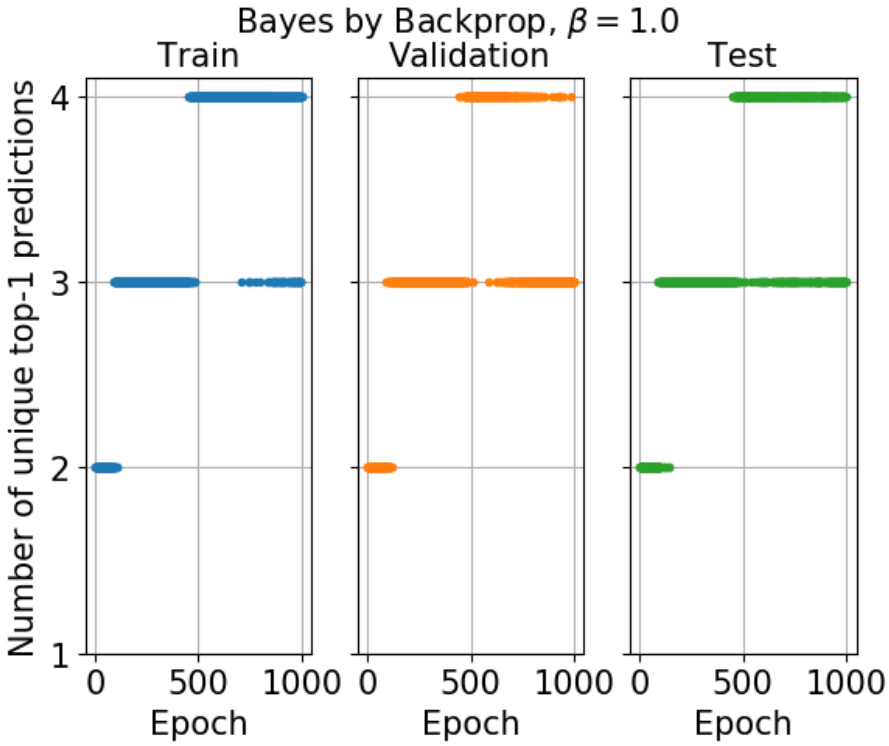
(g) Train **(h)** Validation

(i) Test

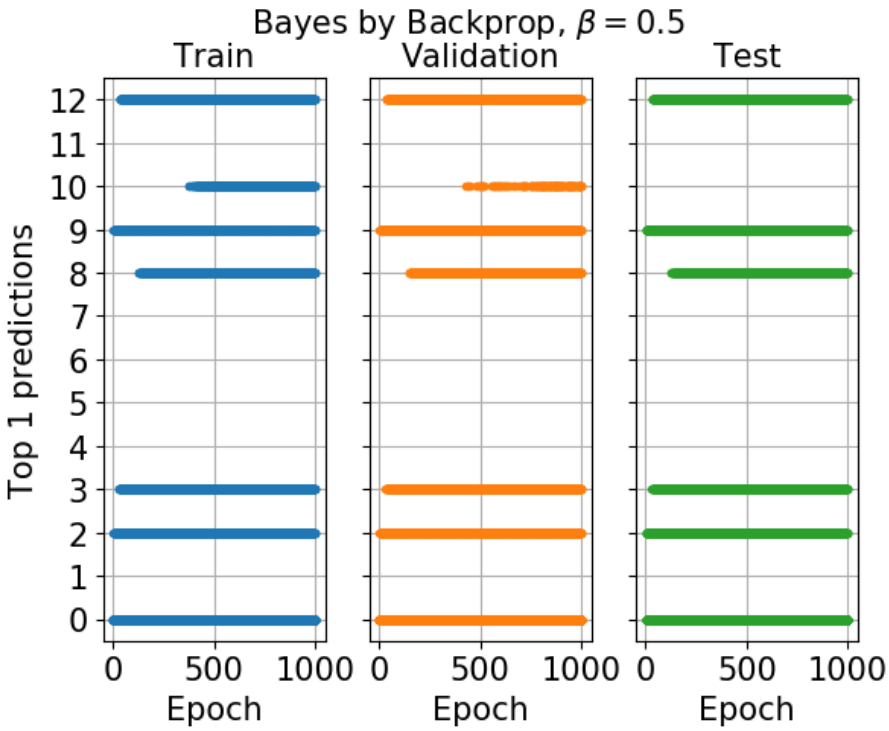
Figure 4.32: The top n accuracy on the first CV fold of the AZ ELN data. Note that for $\beta = 1.0$ the performance is similar on the training, validation and test set. However, for $\beta \in \{0.1, 0.5\}$ the training accuracy is significantly higher. Finally, note that the top 5 accuracy for the validation and test set is similar for all values of β.



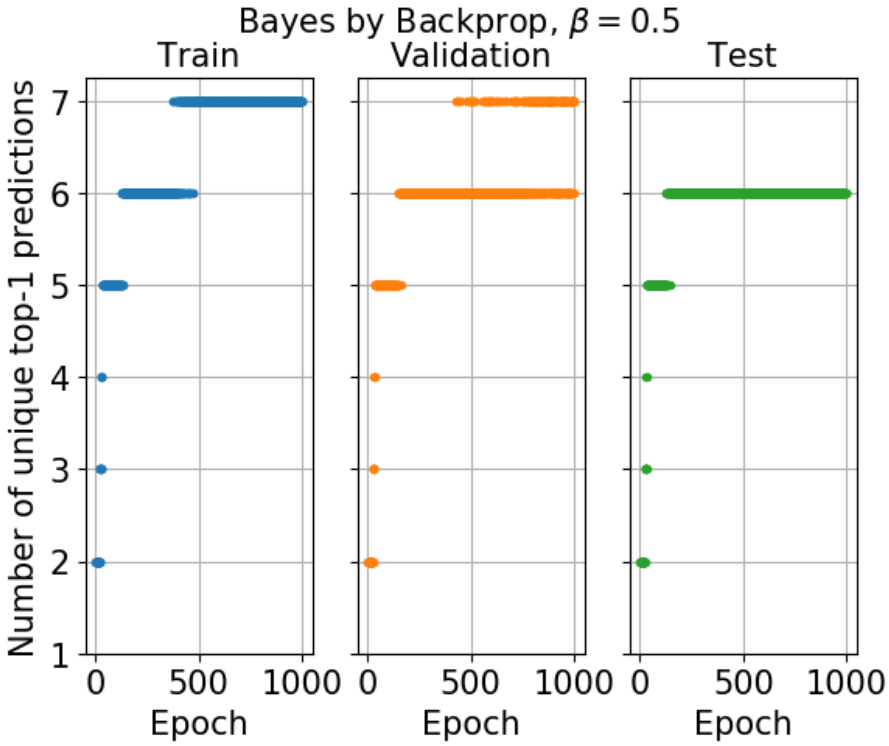
(a) Predicted ligands



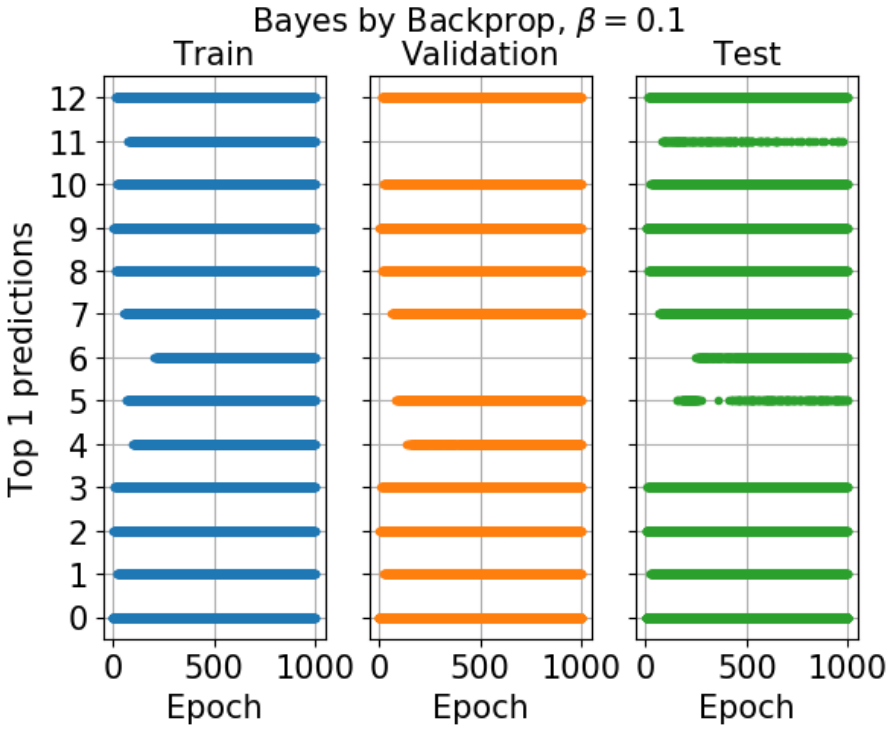
(b) Number of predicted ligands



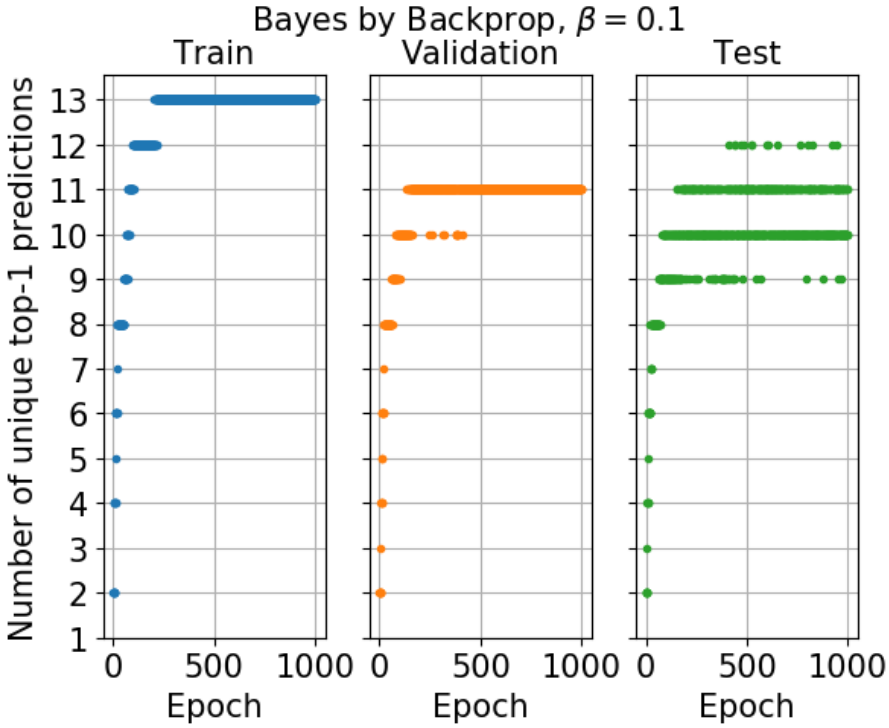
(c) Predicted ligands



(d) Number of predicted ligands



(e) Predicted ligands

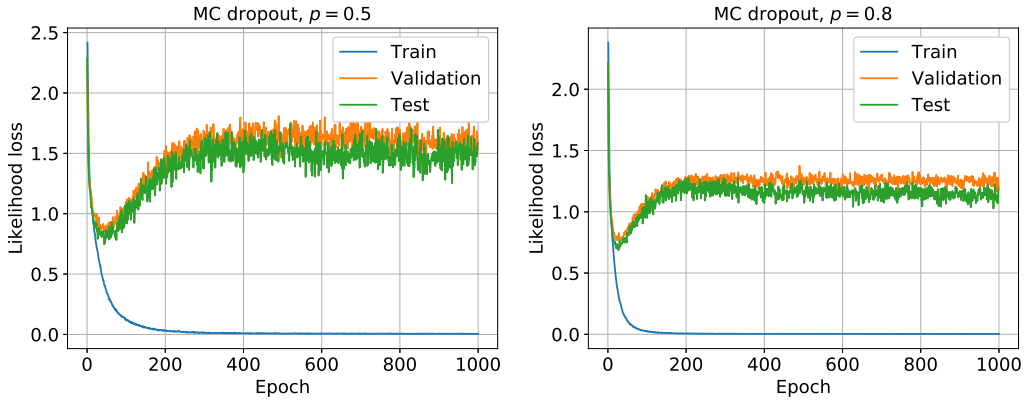


(f) Number of predicted ligands

Figure 4.33: Predicted ligands and number of predicted ligands on the first CV fold of the AZ ELN data. Predicted here means that the ligand is the most frequently observed outcome of the network based on 100 forward passes of the same input. Note that the trend between the three splits is similar for $\beta = 1.0$.

4.2.2 MC Dropout

This section presents the predictive performance of the MC dropout approach obtained on the first CV fold. The results for the second CV fold are presented in Appendix A.2.2, and the remaining folds illustrate similar trends as the second fold. The predictive results obtained for the first fold illustrate less overfitting than the remaining folds. However, the comparison between $p = 0.5$ and $p = 0.8$ presented in this section is consistent for all folds. Finally, recall that p is the probability that a node will be retained.



(a) Likelihood loss (Negative log likelihood)

(b) Likelihood loss (Negative log likelihood)

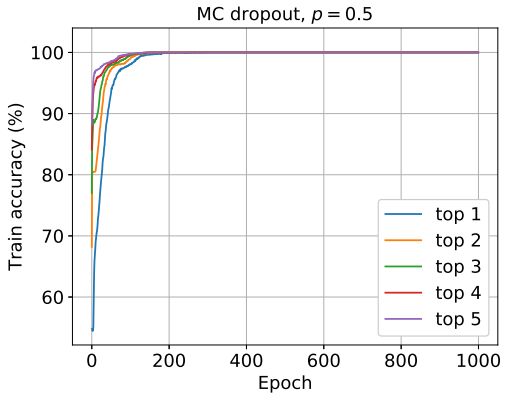
Figure 4.34: The likelihood loss for MC dropout on the first CV fold of the AZ ELN data. The loss is averaged over the number of data points in each set. Note that after the first 50 epochs the model starts to overfit the data for both values of p.

Figure 4.34 illustrates the likelihood loss, i.e., the negative log likelihood of the data given the model, for $p \in \{0.5, 0.8\}$. The loss is averaged over the number of observations in each set. Note that after the initial epochs the validation and test loss increases, which indicates that both models overfit the data. For the $p = 0.8$ model the test loss decreases slightly after the first 200 epochs, which indicates that the model still are learning features that at least generalizes to the test set as well. Moreover, Figure 4.35 visualizes the top n accuracy for the first CV fold of the AZ ELN data of the MC dropout approach. Note that the mode obtains a top 1 accuracy close to 100% on the training split, but approximately 80% for the validation and test split. This discrepancy in performance between the folds further indicates that the models overfit the data. The top 5 accuracy of the validation and test set is similar for both values of p. Furthermore, this trend in top 5 accuracy is similar to the results obtained by the Bayes by Backprop approach, which is displayed in Figure 4.32. Lastly, the fact that the test and validation loss are lower for the $p = 0.8$ model, while the top 1 accuracy is approximately the same, compared to the $p = 0.5$ model, indicates that the former model either is more certain in correct predictions or less certain in incorrect predictions.

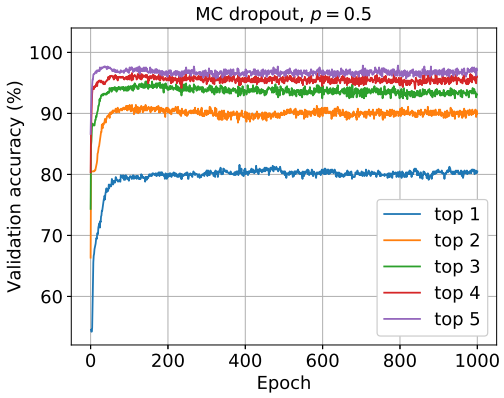
The predictive diversity is visualized in Figure 4.36. Note that both values of p

yields a model that predicts at least 11 out of the possible 13 ligands. This is to be compared to the $\beta \in \{1.0, 0.5\}$ models, see Figure 4.33, where a maximum of 6 and 4 ligands where predicted, respectively. The MC dropout model does also seem to have a slightly better top 1 accuracy than the mentioned Bayes by Backprop models. This might be contributed to the fact that MC dropout manages to predict more ligands.

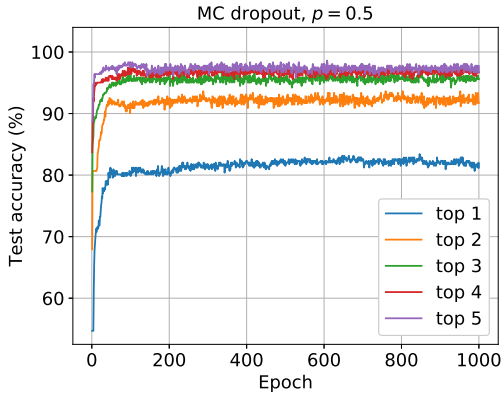
The model with $p = 0.8$ also seems to find slightly more ligands than the $p = 0.5$ model. However, both models struggle to predict ligand 6 and 11 on the validation set, which account for 0.15% and 0.61% of the observations, respectively. The $p = 0.8$ model also predicts more ligands with less training. The fact that the likelihood loss, Figure 4.34, is lower for $p = 0.8$ than $p = 0.5$ also indicates that the model predicts the correct ligand with a larger margin, i.e., the corresponding softmax entry is larger.



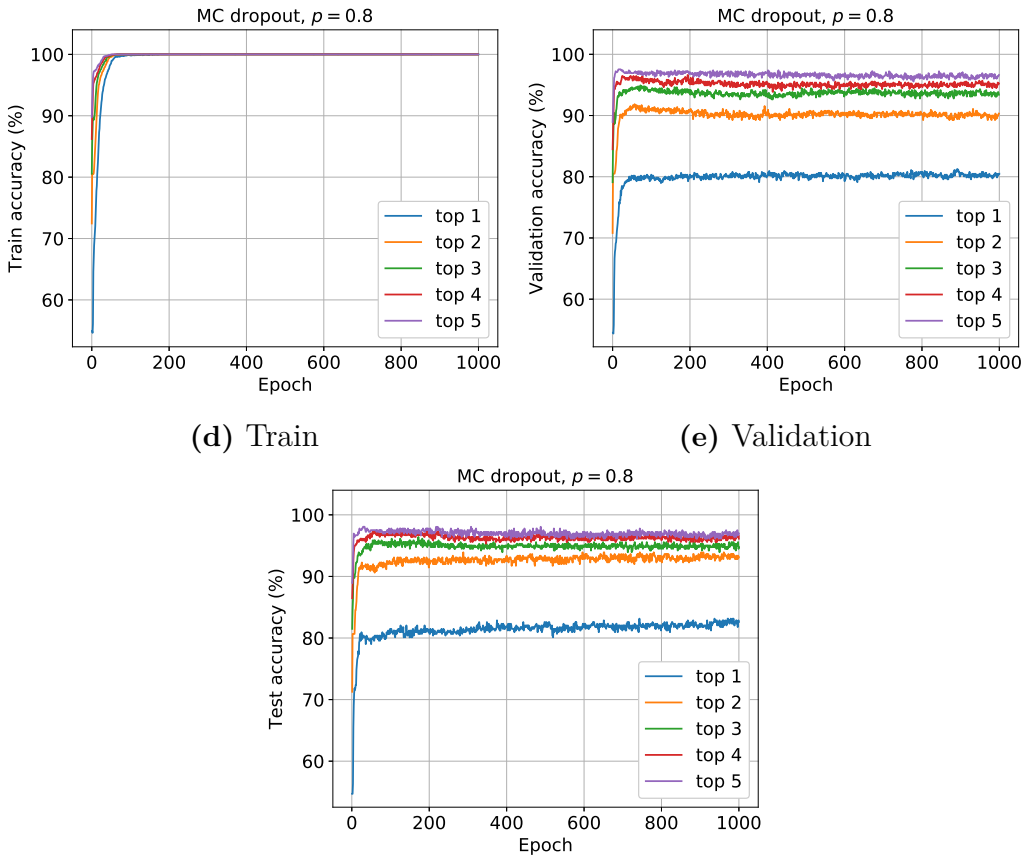
(a) Train



(b) Validation



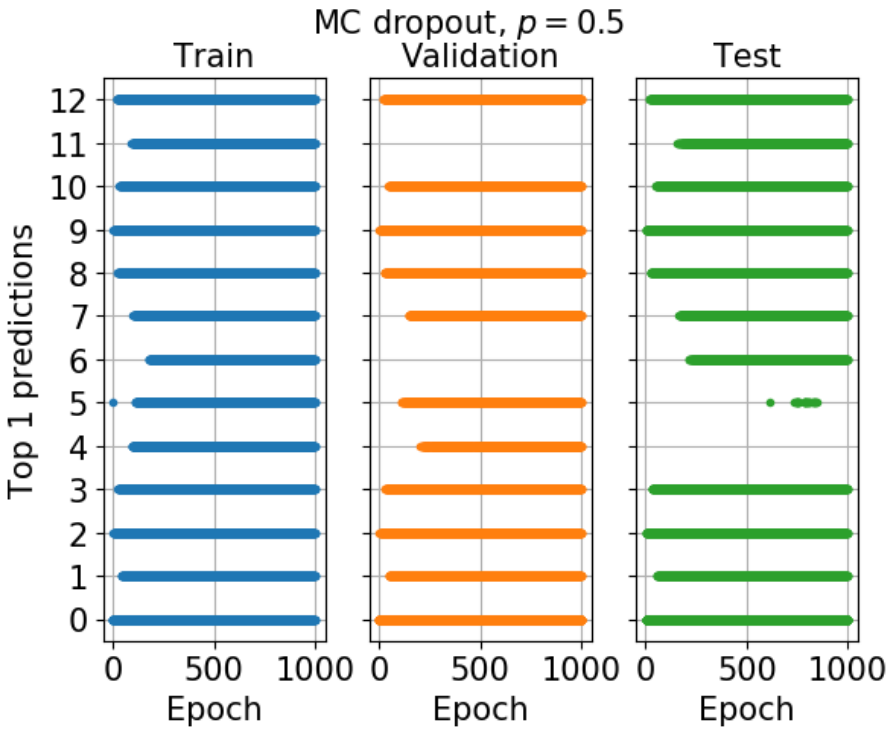
(c) Test



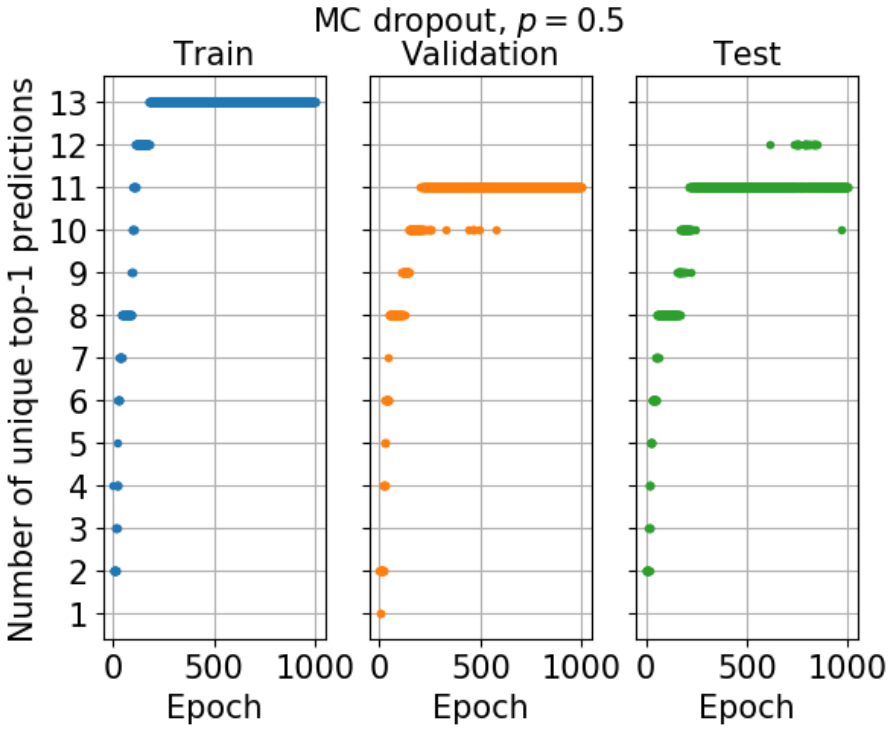
(d) Train **(e)** Validation

(f) Test

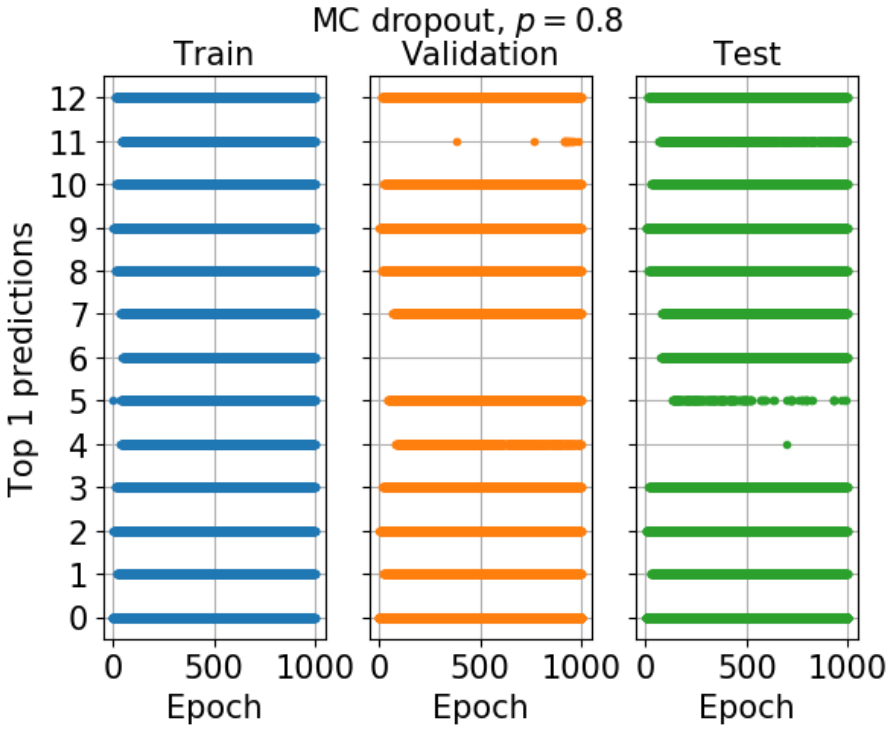
Figure 4.35: The top n accuracy for MC dropout on the first CV fold of the AZ ELN data. Note that for both values of p the top n accuracy for the training set is significantly higher than the corresponding accuracy for validation and test split.



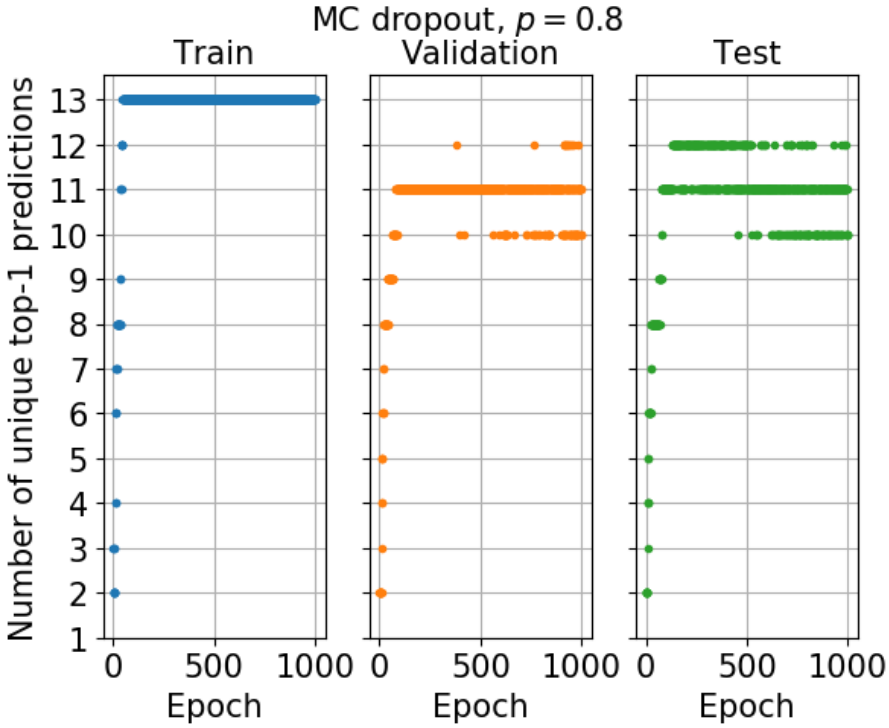
(a) Predicted ligands



(b) Number of predicted ligands



(c) Predicted ligands

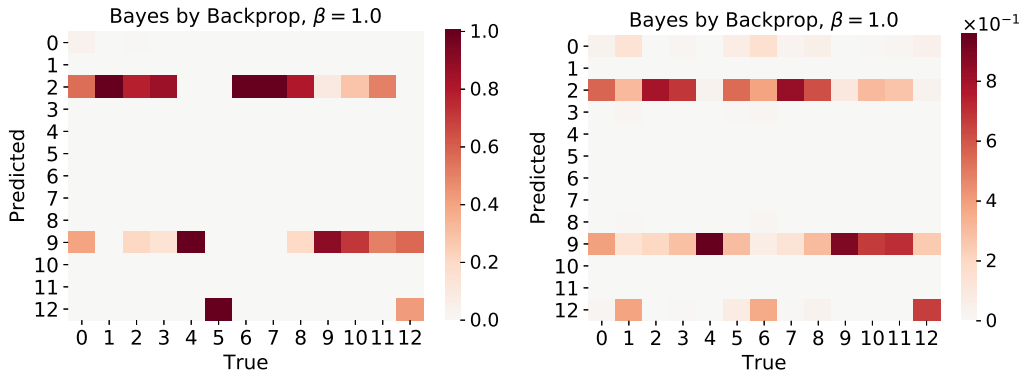


(d) Number of predicted ligands

Figure 4.36: Predicted ligands and number of predicted ligands on the first CV fold of the AZ ELN data. Predicted here means that the ligand is the most frequently observed outcome of the network based on 100 forward passes of the same input. Note that the trend between the validation and test split are similar.

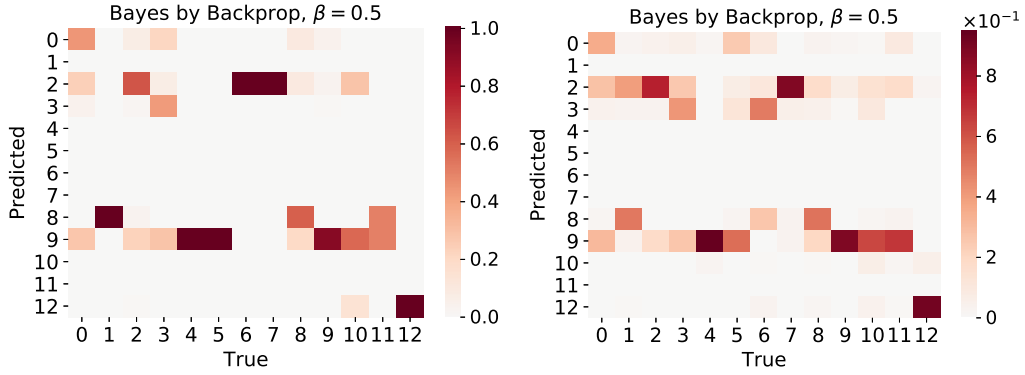
4.3 AZ ELN Uncertainty Evaluation

This section presents the uncertainty of the models, and it is limited to Bayes by Backprop with $\beta \in \{1.0, 0.5\}$ and MC dropout with $p = 0.8$. The other models were discarded since they illustrated inferior performance.



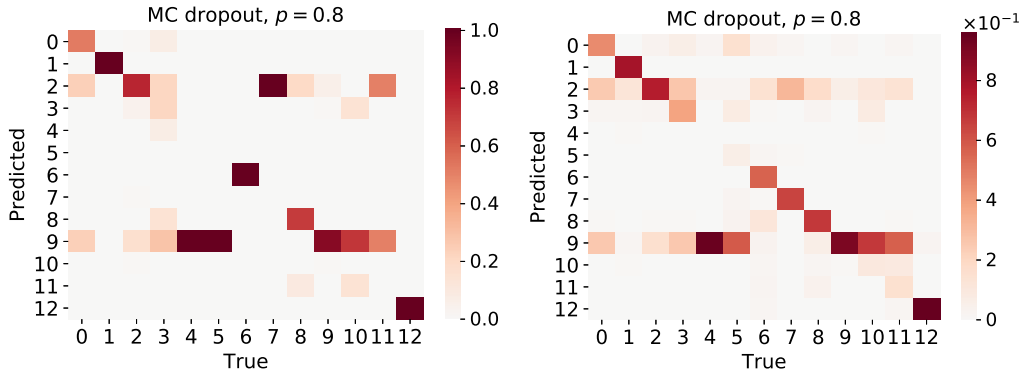
(a) 1 forward pass

(b) 1000 forward passes



(c) 1 forward pass

(d) 1000 forward passes



(e) 1 forward pass

(f) 1000 forward passes

Figure 4.37: The naïve uncertainty, as was described in Algorithm 5, on the test set for Bayes by Backprop with $\beta \in \{1.0, 0.5\}$ and MC dropout with $p = 0.8$. The uncertainty is estimated with 1 and 1000 forward passes. Note that each column represents the true ligand, and that each column is normalized.

Figure 4.37 illustrates the average naïve uncertainty, based on Algorithm 5, of the entire test set. The average is taken w.r.t. all inputs with a specific ligand. The uncertainty was estimated by passing the test set through the network $T \in \{1, 1000\}$ times. The matrix contains elements (Predicted ligand, True ligand), i.e., the rows represent the predicted ligands and the columns represent the true ligands, and each entry corresponds to how often that entry was present in the forward pass of the test set. This means that element $(0, 0)$ corresponds to the fraction by which ligand 0 was deemed the most likely when the true ligand also was ligand 0. The fact that the networks are stochastic is clearly visualized in Figure 4.37 since the matrix element changes when the test set is passed multiple times through the network. That Bayes by Backprop with $\beta = 1.0$ mainly predicts ligand 0, 2, 9 and 12 is consistent with the predicted diversity illustrated in Figure 4.33.

Large diagonal elements are desired since they indicate that the model usually predicts the correct ligand. On the other hand, if a column contains the largest element off the diagonal then the network usually predicts the incorrect ligand for the corresponding reactions. The reaction with ligand 4 illustrate such behaviour for all models since the 4th column consists of an entry close to 1.0 at row 9, i.e., the model mainly predicts ligand 9 even though ligand 4 is correct. Finally, if the network suggest a ligand corresponding to a column with a large uncertainty, then the prediction could be uncertain; this follows from the fact that Figure 4.37 presents the average uncertainty of all inputs with a specific ligand.

In order to evaluate the actual predicted uncertainty of the model, one should pass a single input through the network multiple times. The obtained naïve uncertainty, Algorithm 5, of $T = 1000$ forward passes of a randomly selected input is presented in Table 4.1. Note that the uncertainty obtained from both Bayes by Backprop models contain multiple large elements. This indicates that the models are uncertain about the input. However, it is worth to emphasize that the models seem to be uncertainty whether to classify the input as ligand 2 or 12 for $\beta = 1.0$ and as ligand 2 or 8 for $\beta = 0.5$, and not as ligand 1 which is the correct one. In fact, ligand 1 is not the most likely in any of the 1000 forward passes. MC dropout, on the other hand, illustrates less uncertainty on this input; since it mainly predicts ligand 1.

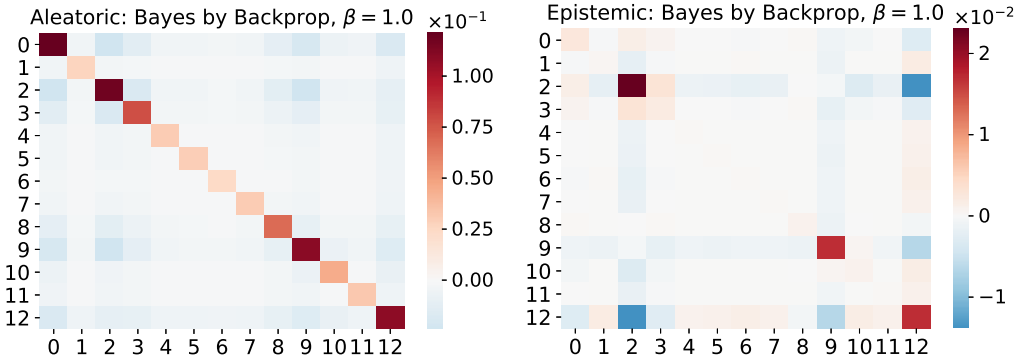
Table 4.1: The naïve uncertainty of Bayes by Backprop with $\beta \in \{1.0, 0.5\}$ and MC dropout with $p = 0.8$. The uncertainty is estimated according to Algorithm 5, and the entropy is computed according to Equation (2.10). Ligand 1 is correct.

Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12	Entropy
$\beta = 1.0$	140	0	308	19	0	0	0	0	9	138	0	0	386	2.33
$\beta = 0.5$	23	0	398	32	0	0	0	0	497	39	1	0	10	2.03
$p = 0.8$	7	796	122	22	10	2	1	4	0	16	11	4	5	1.13

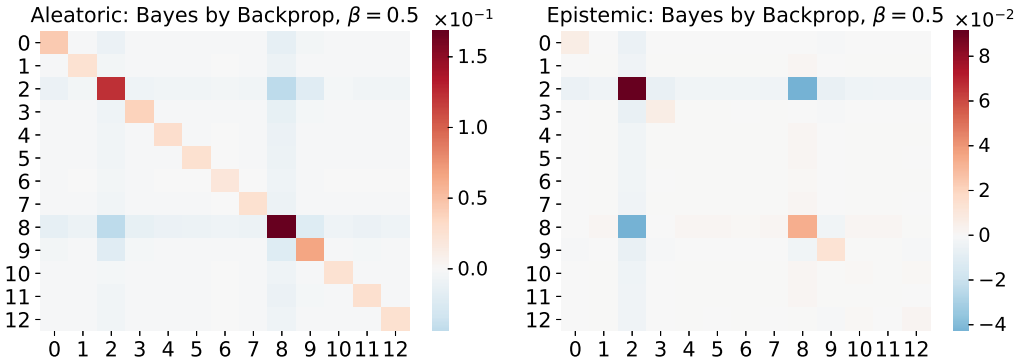
The uncertainty of the input can be further decomposed into aleatoric and epistemic uncertainty according to Equation (2.14). The decomposition is visualized in Figure 4.38, recall that the matrices represent covariance matrices. The total uncertainty of the model is the sum of the aleatoric and epistemic uncertainty. Note that if the matrix contains multiple large entries then there are multiple elements with

Table 4.2: The variational predictive distribution for a random input for Bayes by Backprop with $\beta \in \{1.0, 0.5\}$ and MC dropout with $p = 0.8$. Ligand 1 is correct.

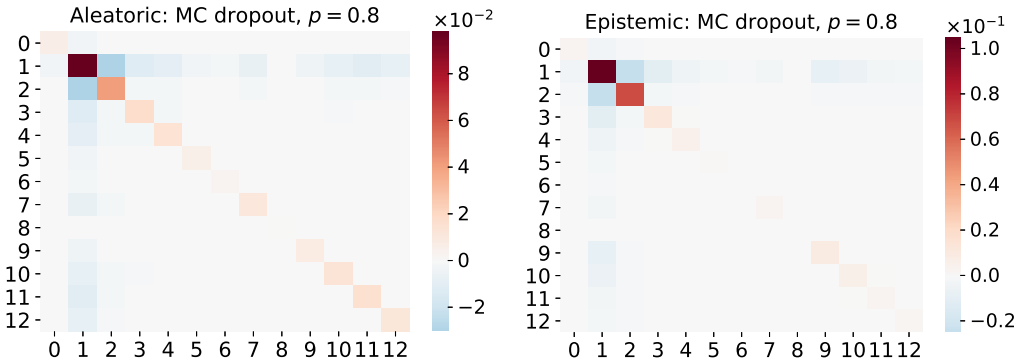
Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12	
$\beta = 1.0$	14	3	17	9	3	3	3	3	7	15	5	3	15	$\times 10^{-2}$
$\beta = 0.5$	5	3	31	5	3	3	2	3	28	9	2	3	3	$\times 10^{-2}$
$p = 0.8$	1	72	13	3	2	1	0	1	0	2	2	2	1	$\times 10^{-2}$



(a) Aleatoric uncertainty **(b)** Epistemic uncertainty



(c) Aleatoric uncertainty **(d)** Epistemic uncertainty



(e) Aleatoric uncertainty **(f)** Epistemic uncertainty

Figure 4.38: The estimated aleatoric and epistemic uncertainty on the same input that was used in Table 4.1 for Bayes by Backprop with $\beta \in \{1.0, 0.5\}$ and MC dropout with $p = 0.8$. The correct ligand is ligand 1.

significant alterations, which could indicate that the model is uncertain. The figure illustrates that both Bayes by Backprop models account most of the uncertainty to the aleatoric uncertainty, i.e., uncertainty which the model can not reduce. That the diagonal elements $(0, 0)$, $(2, 2)$, $(9, 9)$ and $(12, 12)$ are the largest entries for the $\beta = 1.0$ model, and elements $(2, 2)$ and $(8, 8)$ are the largest for the $\beta = 0.5$ model is coherent with the results presented in Table 4.1. This is explained by the large variations between the corresponding predicted ligands from the naïve uncertainty.

MC dropout, on the other hand, account most of the uncertainty to the epistemic uncertainty. This means that the MC dropout model indicates that the uncertainty can be reduced if it was trained with more data. The fact that element $(1, 1)$ is significantly larger for MC dropout indicates that the largest variation in the model is for ligand 1. This is consistent with the results in Table 4.1, where ligand 1 is by far the most likely prediction. Finally, we emphasize that comparing the actual values between the models might be ill-advised since they assume different variational posterior distributions and, thus, are likely to generate different uncertainties.

4.3.1 Uncertainty by Bayes by Backprop With $\beta = 1.0$

The aleatoric and epistemic uncertainty for the input corresponding to the lowest entropy are presented in Figures 4.39a and 4.39b, respectively. The results illustrates that both the aleatoric and epistemic uncertainties are low, i.e., the total uncertainty is low. Note that "low" here is relative to the corresponding element of the variational predictive in Table 4.4. This could indicate that the model is certain about the input. In particular, the low epistemic uncertainty could indicate that the model is certain since this indicates that the predictions do not alter significantly between each forward pass. Moreover, the naïve uncertainty for this input is presented in Table 4.3. The fact that this uncertainty only contains ligand 9 illustrates that the model solely predicts ligand 9 for this input. This further indicates that the model is certain. Finally, the fact that the "Min entropy" observation corresponds to a certain prediction is to be expected since a low entropy indicates that the prediction does not enclose much information.

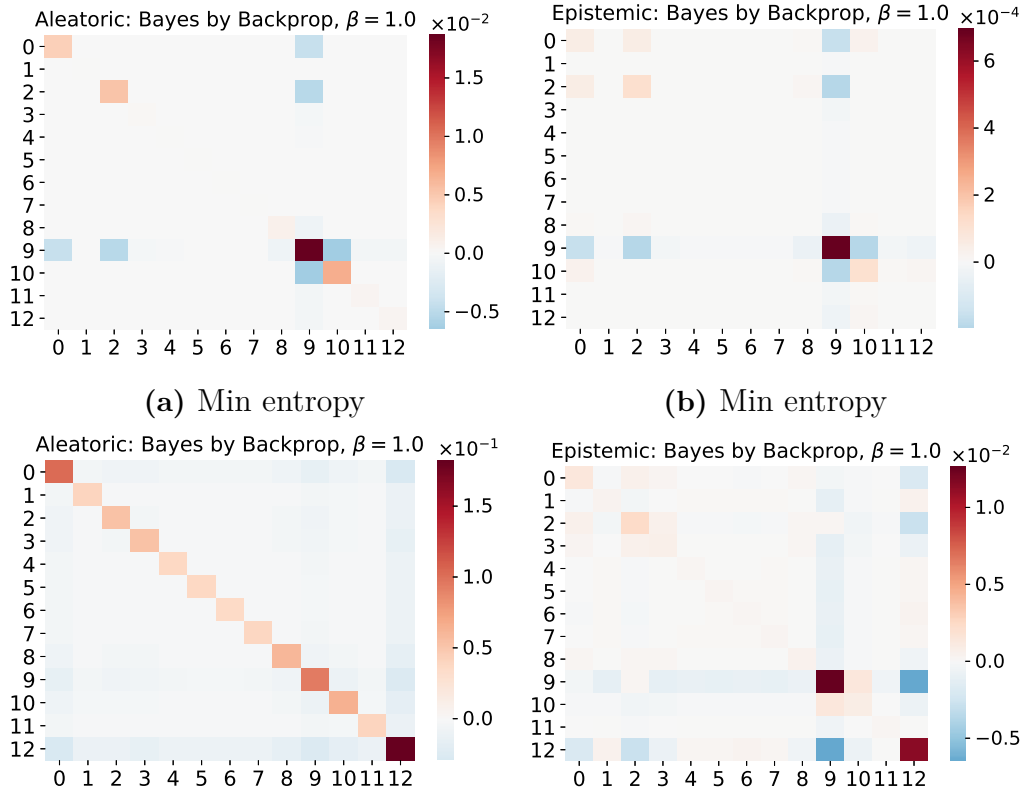
Figures 4.39c and 4.39d visualize the estimated uncertainty for the input with the largest entropy. The results illustrates that the model contains multiple large diagonal elements. This could indicate that the model is uncertain about how to classify the input, since there are multiple softmax entries which varies significantly between forward passes. The naïve uncertainty is presented in Table 4.3. The fact that this uncertainty mainly predicts ligand 12 could indicate that the model is certain. However, from the variational predictive in Table 4.4 it is clear that even though ligand 12 is deemed the most likely in the majority of the forward passes, the corresponding probability is quite low. Furthermore, note that the maximum epistemic uncertainty for the "Max entropy" input is more than one order of magnitude larger than the corresponding value for the "Min entropy" input. This indicates that lower values of the epistemic uncertainty could correspond to more certain inputs. Lastly, that the entropy of the "Max entropy" observation is close to the maximum possible entropy, which is $\log(13)$ since there are 13 different ligands, indicates that the observation contains much information; which can be interpreted as an uncertain observation.

Table 4.3: The naïve uncertainty of Bayes by Backprop with $\beta = 1.0$. The inputs correspond to the minimum and maximum entropy, the entropy is based in the variational predictive which is presented in Table 4.4. The true ligand for the min and max entropy input is ligand 9 and ligand 12, respectively.

Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12	Entropy
Min entropy	0	0	0	0	0	0	0	0	0	1000	0	0	0	0.13
Max entropy	55	0	21	2	0	0	0	0	0	141	1	0	780	2.34

Table 4.4: The variational predictive distribution for the inputs that correspond to the minimum and maximum entropy. The distribution is estimated by Bayes by Backprop with $\beta = 1.0$. The true ligand for the min and max entropy input is ligand 9 and ligand 12, respectively.

Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12	
Min entropy	0	0	1	0	0	0	0	0	0	98	1	0	0	$\times 10^{-2}$
Max entropy	12	4	6	6	4	4	4	4	7	12	7	4	26	$\times 10^{-2}$



(a) Min entropy

(b) Min entropy

(c) Max entropy

(d) Max entropy

Figure 4.39: The estimated aleatoric and epistemic uncertainty on the inputs that corresponds to the minimum and maximum entropy. The true ligand is ligand 9 and 12 for the min and max entropy observation, respectively.

4.3.2 Uncertainty by Bayes by Backprop With $\beta = 0.5$

The aleatoric and epistemic uncertainty for the input corresponding to the lowest entropy are presented in Figures 4.40a and 4.40b, respectively. The results illus-

trates that both the aleatoric and epistemic uncertainties are low, i.e., the total uncertainty is low. Note that "low" here is relative to the corresponding element of the variational predictive in Table 4.6. This could indicate that the model is certain about the input. In particular, the low epistemic uncertainty could indicate that the model is certain since this indicates that the predictions do not alter significantly between each forward pass. Moreover, the naïve uncertainty for this input is presented in Table 4.5. The fact that this uncertainty only contains ligand 9 illustrates that the model solely predicts ligand 9 for this input. This further indicates that the model is certain. Finally, the fact that the "Min entropy" observation corresponds to a certain prediction is to be expected since a low entropy indicates that the prediction does not enclose much information.

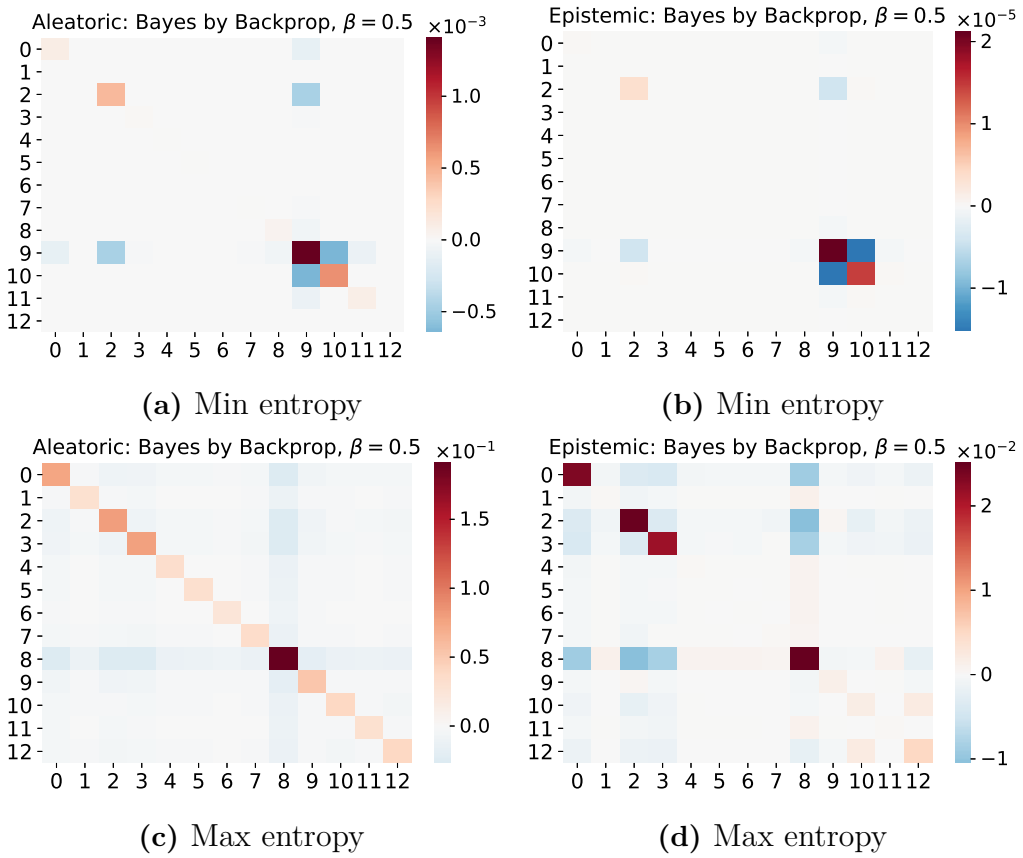
Figures 4.40c and 4.40d visualize the estimated uncertainty for the input corresponding to the largest entropy. The epistemic uncertainty contains multiple large diagonal elements, however, these elements are smaller than element $(8, 8)$ of the aleatoric uncertainty. In particular, this element is significantly larger than all other elements of both the aleatoric and epistemic uncertainty. This could indicate that the model is rather certain about the input since the main alterations are related to ligand 8, and the remaining variations are comparatively small. The variational predictive in Table 4.6 illustrates that ligand 8 is the ligand that most frequently correspond to the largest softmax entry. However, the corresponding probability of 0.32 is quite low compared to the estimated aleatoric and epistemic uncertainty (Figures 4.40c and 4.40d, respectively). This could indicate that the model is uncertain about the input. The variance of the naïve uncertainty, Table 4.5, also indicates that the model is uncertain, even though ligand 8 is the most likely output in the majority of the forward passes. Lastly, that the entropy of the "Max entropy" observation is close to the maximum possible entropy, which is $\log(13)$ since there are 13 different ligands, indicates that the observation contains much information; which can be interpreted as an uncertain observation.

Table 4.5: The naïve uncertainty of Bayes by Backprop with $\beta = 0.5$. The inputs correspond to the minimum and maximum entropy, the entropy is based in the variational predictive which is presented in Table 4.6. The true ligand for the min and max entropy input is ligand 9 and ligand 8, respectively.

Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12	Entropy
Min entropy	0	0	0	0	0	0	0	0	0	1000	0	0	0	0.01
Max entropy	106	0	114	118	0	0	0	0	617	6	4	0	35	2.22

Table 4.6: The variational predictive distribution for the inputs that correspond to the minimum and maximum entropy. The distribution is estimated by Bayes by Backprop with $\beta = 0.5$. The true ligand for the min and max entropy input is ligand 9 and ligand 8, respectively.

Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12	
Min entropy	0	0	0	0	0	0	0	0	0	100	0	0	0	$\times 10^{-2}$
Max entropy	11	3	12	11	3	3	3	4	32	6	4	3	5	$\times 10^{-2}$



(a) Min entropy

(b) Min entropy

(c) Max entropy

(d) Max entropy

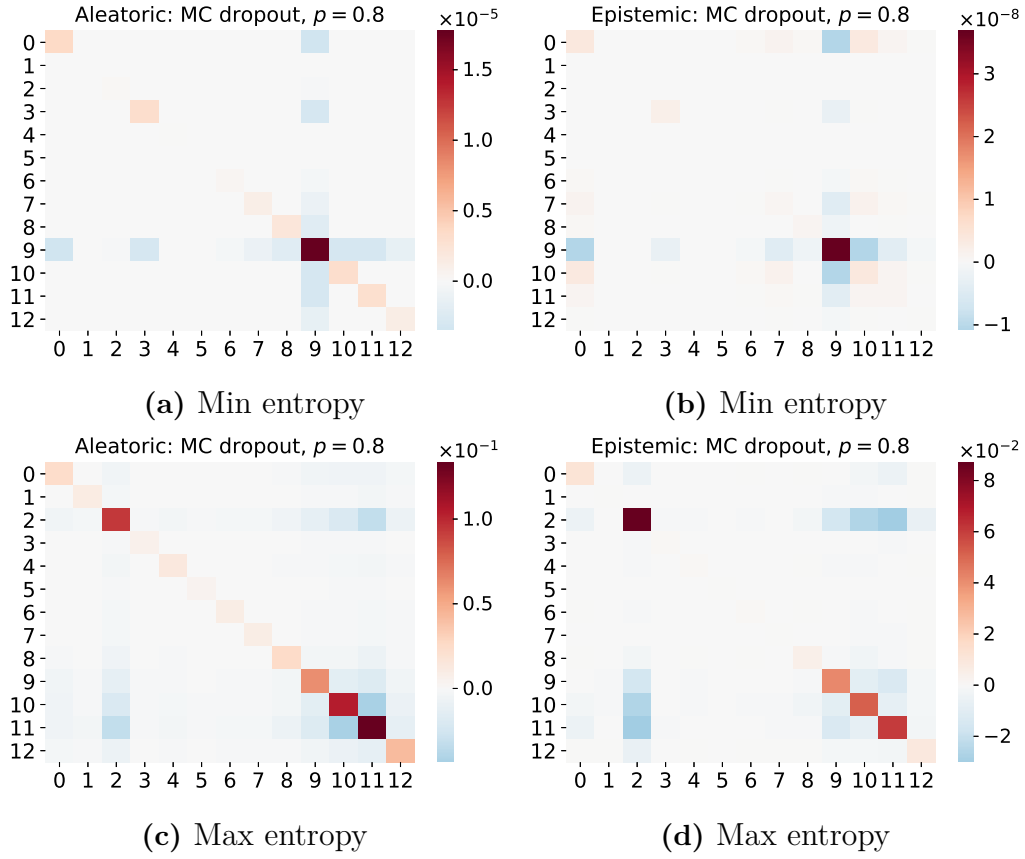
Figure 4.40: The estimated aleatoric and epistemic uncertainty for Bayes by Backprop with $\beta = 0.5$ on the inputs that corresponds to the minimum and maximum entropy. The true ligand is ligand 9 and 8 for the min and max entropy observation, respectively.

4.3.3 Uncertainty by MC Dropout With $p = 0.8$

The aleatoric and epistemic uncertainty for the input corresponding to the lowest entropy are presented in Figures 4.41a and 4.41b, respectively. The results illustrates that both the aleatoric and epistemic uncertainties are low, i.e., the total uncertainty is low. Note that "low" here is relative to the corresponding element of the variational predictive in Table 4.8. This could indicate that the model is certain about the input. In particular, the low epistemic uncertainty could indicate that the model is certain since this indicates that the predictions do not alter significantly between each forward pass. Moreover, the naïve uncertainty for this input is presented in Table 4.7. The fact that this uncertainty only contains ligand 9 illustrates that the model solely predicts ligand 9 for this input. This further indicates that the model is certain. Finally, the fact that the "Min entropy" observation corresponds to a certain prediction is to be expected since a low entropy indicates that the prediction does not enclose much information.

Figures 4.41c and 4.41d visualize the estimated uncertainty for the input with the largest entropy. The results illustrates that the model contains multiple large diagonal elements. In particular, element $(2, 2)$ and $(11, 11)$ are the two largest elements of

the matrices. This could indicate that the model is uncertain about how to classify the input. Moreover, the naïve uncertainty for this input is presented in Table 4.7. The fact that this uncertainty contains multiple large elements illustrates that the predictions of the model alter between the forward passes. This indicates that the model is uncertain. Both the decomposition and the naïve uncertainty illustrate that the model mainly is uncertain about whether the correct ligand is 2 or 11. Furthermore, note that the maximum epistemic uncertainty for the "Max entropy" input is multiple orders of magnitude larger than the corresponding value for the "Min entropy" input. This indicates that lower values of the epistemic uncertainty could correspond to more certain inputs. Lastly, that the entropy of the "Max entropy" observation is larger than the entropy for the "Min entropy" observation indicates that a large entropy could correspond to a more uncertain observation.



(a) Min entropy **(b)** Min entropy

(c) Max entropy **(d)** Max entropy

Figure 4.41: The estimated aleatoric and epistemic uncertainty for MC dropout with $p = 0.8$ on the input that corresponds to the minimum and maximum entropy. The true ligand is ligand 9 and 11 for the min and max entropy input, respectively.

Table 4.7: The naïve uncertainty of MC dropout with $p = 0.8$. The inputs correspond to the smallest and largest entropy, the entropy is based in the variational predictive which is presented in Table 4.8. The true ligand for the min and max entropy input is ligand 9 and ligand 11, respectively.

Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12	Entropy
Min entropy	0	0	0	0	0	0	0	0	0	1000	0	0	0	0.00
Max entropy	31	1	279	4	1	0	2	2	16	135	191	305	33	1.94

Table 4.8: The variational predictive distribution for the inputs that correspond to the smallest and largest entropy. The distribution is estimated by MC dropout with $p = 1.0$. The true ligand for the min and max entropy input is ligand 9 and ligand 11, respectively.

Ligand	0	1	2	3	4	5	6	7	8	9	10	11	12	
Min entropy	0	0	0	0	0	0	0	0	0	100	0	0	0	$\times 10^{-2}$
Max entropy	4	1	24	1	2	1	1	1	3	12	19	26	5	$\times 10^{-2}$

5

Discussion

5.1 Active Learning Using Uncertainty Quantification

This section discusses the results of the prediction of yields for different reaction components when using different strategies to expand the training set. Recall that reaction components denote both the reaction conditions and the reactants of a reaction. Five different comparisons obtained on the Merck and Pfizer data are discussed:

1. (absolute) uncertainty sampling without correlation simulations compared to relative uncertainty sampling without correlation simulations
2. uncertainty sampling compared to uncertainty sampling without correlation simulations
3. uncertainty sampling without correlation simulations compared to random sampling
4. Kennard-Stone Initialization: uncertainty sampling without correlation simulations compared to sampling using Kennard-Stone algorithm when both strategies were initialized using Kennard-Stone algorithm
5. uncertainty sampling without correlation simulations vs random sampling when both strategies utilized fingerprints as side information to Macau

Comparisons 1, 2, 3 and 5 used random initialization. Recall that "(absolute) uncertainty" is used unless otherwise stated.

5.1.1 (Absolute) Uncertainty Compared to Relative Uncertainty

As shown in Figures 4.1, 4.3 and 4.2, (absolute) uncertainty sampling yields both a lower RMSE and predictive variability compared to relative uncertainty sampling. Also, as displayed in Figure 4.4, they show a similar trend in the PR AUC score. Moreover, Figure 4.5 illustrates how the methods select points in different ways based on the observed yields.

The objective with this comparison was to investigate if relative uncertainty could provide any additional information that could be useful to active learning compared

to (absolute) uncertainty. The results indicate that relative uncertainty provides different information, compared to (absolute) uncertainty sampling, since relative uncertainty sampling seems to more emphasize the selection of points with low observed yield. Instead, (Absolute) uncertainty seems to emphasize the selection of points with high or intermediate yield. This is to be expected since points can have a large variance merely due to the fact that the associated yield is high. However, (absolute) uncertainty does not incorporate information about the associated yield into the uncertainty, thereby it is inclined to select points with higher yield earlier than relative uncertainty. An interesting observation is that the extra information does not result in better RMSE. This might be due to the fact that points with lower yield also have a lower RMSE since their average is lower, thereby reducing their deviations. The results indicate that (absolute) uncertainty performs better than relative uncertainty, both with respect to variance and absolute error. The relative error plotted against the observed yield, as seen in Figure 4.6, demonstrates a lower relative error of points with an observed yield lower than 20% when using relative uncertainty compared to (absolute) uncertainty. Relative uncertainty sampling does not indicate any better performance on the points with an observed yield greater than 20%. However, improving the relative error of points with low yield could is advantageous when exploring whether a reaction is successful or not, i.e., having a sufficiently high yield. In this thesis a reaction is successful if it has a yield greater than or equal to 5% but this threshold can vary depending on the purpose of the reaction.

A possible explanation to the better performance of (absolute) uncertainty sampling, with respect to the absolute error and variance, could be that it is more difficult for the model to predict successful reactions. However, the PR AUC scores, as seen in Figure 4.4, show no clear evidence for this. On the other hand, as discussed regarding Figure 4.6, the better performance (based on the relative error) of (absolute) uncertainty sampling seems to be for points with observed yields greater than 20%. Hence, (absolute) uncertainty seems to be the better choice when overall better performance is needed, while it can be preferable to use when it is of importance to accurately predict observed yields lower than 20%.

5.1.2 Uncertainty With and Without Correlation Simulations

Correlation simulations were based on the idea that it might be suboptimal to add multiple points which enclose the same information, compared to solely adding points that provide new information to the training set. In particular, the way the correlation simulations were implemented, see Section 3.1.3, it should be more consistent in decreasing the variance. This would then give a less greedy algorithm compared to sampling without correlation simulations, since sampling without correlation simulations add points solely based on the variance. This means that sampling without correlation simulations does not consider that the variance can decrease when new points are added, but simply add the points associated with the largest variance. However, if the variance of a point x decreases when new points are added, then that indicates that x could be correlated with the already added

points. Thus, it might be better to add another point $\widetilde{x}$ whose variance did not decrease, even though the variance of $\widetilde{x}$ might be lower than the variance of x.

As displayed in Figures 4.7 and 4.9a, the correlation simulations did not improve the performance with respect to the absolute error in the predictions. In fact, correlation simulations seems to give worse performance with respect to absolute error, i.e., the RMSE, in beginning. A better performance with respect to absolute error was not expected since the objective of the method rather is to consistently decrease the variance (or uncertainty) and not the absolute error. On the other hand, a worse performance in the beginning was unexpected. In fact, by using a less uncertain model one could hope for a lower error, but this also depends on the calibration of the model, i.e., how well certain predictions are correlated with correct predictions. However, the results in Figures 4.8 and 4.9b give no clear indications that uncertainty sampling with correlation simulations yields a lower variance compared to uncertainty sampling without correlation simulations. Some folds illustrate a lower mean predictive variability in the beginning when utilizing correlation simulations but other folds illustrate the opposite behaviors in the beginning. This is illustrated when comparing Fold 1 and 2 in Figure 4.8. What seems to be consistent between the folds is that after a sufficient number of iterations the difference between the variabilities of the strategies is close to zero and varies between being positive and negative. This would then indicate that, after a sufficient number of data points in the training set, the performance with respect to the uncertainty (variance) is similar. This means that the implemented correlation simulations do not seem to have any significant effect on the results. The similar performance in the last iterations could be a reflection of the fact that both strategies seem to add similar points with respect to the observed yield, as illustrated in Figure 4.11.

We still believe that correlation simulations can be an important part if one wants to use uncertainty for active learning in Bayesian models. The approach in this thesis tries to simulate the correlations of a group of points, instead of simulating correlations for each point separately. This simplification was utilized due to the time associated with separate correlation simulations. Figure 5.1 shows the differences

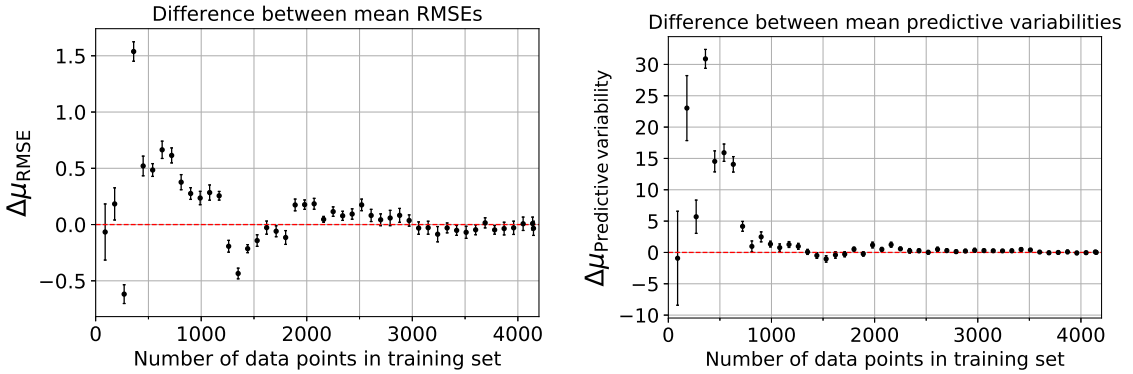
$$\Delta\mu_{\text{predictive variability}} = \mu^{N_c=1}_{\text{predictive variability}} - \mu^{N_c=5}_{\text{predictive variability}},$$

and

$$\Delta\mu_{\text{RMSE}} = \mu^{N_c=1}_{\text{RMSE}} - \mu^{N_c=5}_{\text{RMSE}}$$

between uncertainty sampling with the correlation simulation sizes $N_c = 1$ and $N_c = 5$ as a function of the number of points from fold 9 of the Pfizer data that have been added to the training set. See Section 3.1.3 to recall the meaning of the correlation simulation sizes in the uncertainty sampling. The differences in the figure give no indications on any significant difference of using $N_c = 5$ (adding 5 points before doing correlation simulations) compared to simulating correlations for each point, except for the first iterations. Therefore, the results in this thesis could be a mere illustration of the difficulties to simulate correlations between points, since the size of N_c did not appear to have a significant impact on the results. However, we did not investigate the accuracy of the simulations, but rather if they had any

significant effect on the sampling. Therefore, it would be interesting to both explore other ways to simulate correlations and to investigate how accurate the simulations are.



(a) RMSE **(b)** Predictive variability

Figure 5.1: The differences $\Delta\mu_{\text{predictive variability}}$ and $\Delta\mu_{\text{RMSE}}$ between uncertainty sampling with the correlation sampling sizes $N_c = 1$ and $N_c = 5$ as a function of the number of points from fold 9 of the Pfizer data that have been added to the training set. The red dotted line displays where the difference is equal to zero and the error bars display the 95% approximate confidence interval.

5.1.3 Uncertainty Compared to Random

Figures 4.12, 4.13 and 4.14 indicate that uncertainty sampling performs better with respect to absolute error (RMSE) and variance compared to random sampling when there is a sufficient number of points in the training set. Note that the average difference in RMSE is always greater than -1.5. A maximum negative difference of -1.5 yields a greater negative difference than the difference that was shown for the comparison between (absolute) uncertainty compared to relative uncertainty, see Figure 4.3a. When conducting a comparison between a method with information about the uncertainty in the predictions and a method without any information, one would expect the method with information to perform significantly better. Lastly, it is worth noting that even though the difference in RMSE is quite low, the difference compared to the RMSE of the two methods indicates that uncertainty sampling is around 5% better than random sampling.

As seen in Figure 4.17, the added observed yields in each iteration seem to differ between the strategies. This is expected since random should give a good diversity of observed yields in every iteration, while the observed yields with the highest uncertainty can vary between each iteration. When comparing the added observed yields of uncertainty sampling and random sampling, the variations in the uncertainty of points with different observed yields between iterations seem to be visible. Also, the two-sided p-values from the Mann-Whitney U test in Figure 4.18 show that there is a significant difference between the two training sets when sufficient number of points has been added to the training. This is due to the fact that the test demonstrates p-values lower than 0.05. It is expected that there is no significant difference in the

beginning and the end since both utilize the same random initialization and only a few points will differ between the sets in the end. Since this shows a significant difference between the training sets of the sampling strategies, it seems like uncertainty sampling finds a different training set that shows a slightly better, but still similar, performance. The fact that the training sets are significantly different but still show similar performance could actually be an indication that Macau is able to find relevant relationships in the data sets. In fact, this could indicate that Macau is a suitable model for chemical predictions, rather than that uncertainty sampling and random sampling are two suitable methods for expanding the training data set.

A deficit with uncertainty sampling is that when the data is very sparse then the model fails to reliably predict where more data is needed. This is illustrated by the superior performance of random sampling in the beginning, e.g., Figure 4.14a. It could, therefore, be advantageous to use random sampling in the beginning (in more iterations than just the initialization), since this method collects points which the model is both certain and uncertain about. The diversity of points in the training set could then potentially provide valuable information for the model to better estimate the uncertainty of new points and thereby more reliably indicate where more data is needed. However, due to practical limitations such as limited resources, one would maybe like to make predictions with only a few training points. Then we would not add enough points to the training set to make use of the benefit of uncertainty sampling and, therefore, it could be advantageous to only use random sampling. When predicting unknown data, usually only the training error is available, and one would then want to easily determine when it is beneficial to use uncertainty sampling. Thus, a simple measure or indicator on when to use uncertainty sampling is needed for the method to be truly useful; this is an area for future research.

The fact that uncertainty sampling eventually outperforms random sampling indicates that uncertainty-based active learning potentially could be a useful method to extend the data. In particular, in chemical synthesis, uncertainty-based active learning could be used to indicate which experiments to conduct more efficiently than conducting random experiments. This means that the time and cost associated with synthesizing hit compounds can be reduced by utilizing the uncertainty-based active learning approach to explore the chemical space. However, it is worth noting that the method investigated here is limited to a case where only the associated yield of a reaction is of interest, since what compound that is formed in the reaction is not incorporated in the output of the matrix factorization. Despite this limitation, the results indicate that the method potentially could be an important step towards a fully autonomous synthesis lab; since the method provides a more efficient exploration of what experiments to conduct than random exploration, i.e., uncertainty-based active learning seems to be a smarter way to explore the space than random sampling. In particular, the fact that uncertainty-based active learning seems to select points with high yield is beneficial since in synthesis prediction a high yield is desirable. This is true since synthesis generally consists of a sequence of reactions, therefore, a high yield in each step is essential in order to obtain a viable sequence. However, we believe that our work does not provide enough investigation of uncertainty-based active learning for synthesis prediction so that it can replace a

chemist. Thus, if uncertainty-based active learning is to be used in an autonomous setting then further research is needed.

5.1.4 Uncertainty Compared to Kennard-Stone Algorithm

Figures 4.19, 4.20 and 4.21 indicate a significantly better performance, with respect to RMSE and variance, for uncertainty sampling compared to the Kennard-Stone algorithm when a sufficient number of points have been added to the training set. Also, the similar trend is visualized in the area under precision-recall curve (PR AUC), as seen in Figure 4.22. The behavior and magnitudes of the differences are similar to what was demonstrated in the comparison between uncertainty sampling and random sampling, as seen in the results in both Section 4.1.3.1 and Appendix A.1.3.1. Also, as expected, the observed yields of the points that were added in each iteration of the Kennard-Stone algorithm (see Figure 4.23b) is similar to random sampling (see Figure 4.17b) since the dissimilarity measure implies a large number of ties, which are resolved randomly. Thus, they show the same difference to the added observed yields of uncertainty sampling as random sampling does. The major difference between random sampling and the Kennard-Stone algorithm seems to be that the Kennard-Stone algorithm displays a higher PR AUC score, as seen when comparing Figures 4.16 and 4.22. This is expected to some extent since the fact that Kennard-Stone algorithm adds points with a greater diversity of different choices of reaction components should imply an improved diversity of successful and unsuccessful reactions.

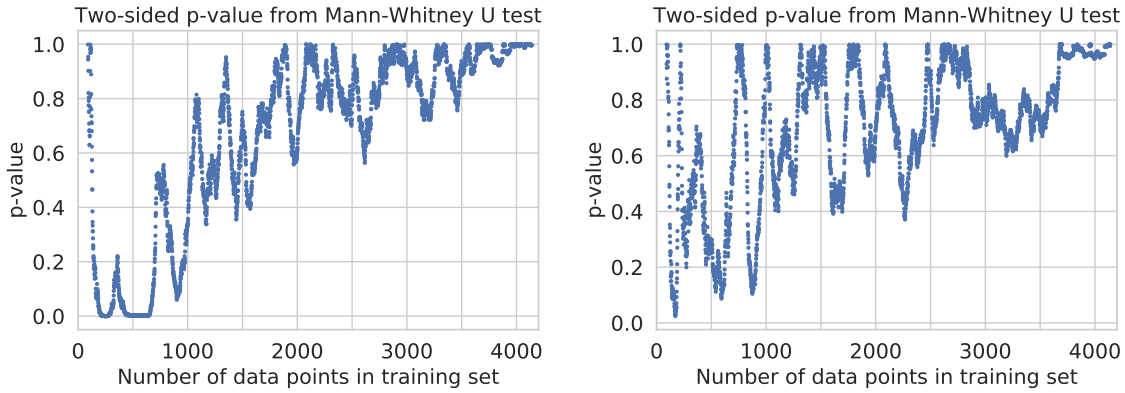
This study was merely an attempt to compare uncertainty sampling with a design of experiments approach and see the effect of a nonrandom initial training set. As mentioned above, the results seem to be similar to the results of random sampling. This is most likely due the similarity between the Kennard-Stone algorithm and random sampling since the ties are resolved by randomly picking points from the set of ties. The limitations that are causing this similarity is the data and the used dissimilarity measure (i.e., the measure of distance used in the Kennard-Stone algorithm), which creates a lot of ties between the points in the data. In particular, the dissimilarity measure. The ties arise because the dissimilarity measure is discrete and simply counts how many reaction components that are different. That is, the dissimilarity measure is limited by the number of reaction components. With fewer ties, the Kennard-Stone algorithm should perform less similarly to random, at least in the way points are selected. In order to use a different measure of distance one could use different chemical descriptors to represent the different choices of reaction components which can give a continuous measure or one could find a better dissimilarity measure. This could probably improve its performance. Therefore, it would be interesting to further investigate how the dissimilarity measure affects the Kennard-Stone initialization and the sampling based on the Kennard-Stone algorithm.

5.1.5 Uncertainty Compared to Random When Utilizing Fingerprints

The objective of using fingerprints was to investigate if, and in what way, additional chemical information would affect the comparison between uncertainty sampling without correlation simulations and random sampling. Fingerprints are described in Section 2.7.1 and how they are incorporated in Macau is described in Section 3.1.1. Utilizing fingerprints seems to demonstrate results (see Section 4.1.5) similar to the results when fingerprints were not used, which are presented and discussed in the previous Sections 4.1.3 and 5.1.3. Here we focus on discussing the major differences in performance between utilizing fingerprints and not utilizing fingerprints.

The PR AUC scores in Figure 4.28, when using fingerprints, shows a slight improvement compared to the PR AUC scores in Figure 4.16, which do not utilize fingerprints. For all folds, the PR AUC score seems to obtain higher overall values when utilizing fingerprints. Also, using fingerprints seems to result in a slightly lower RMSE, as seen when comparing mean RMSE of fold 1 across Figures 4.15a and 4.27a. Hence, this additional information seems to improve Macau's absolute error and its ability to predict if a reaction is successful or not, i.e., predict if a reaction has a yield greater or equal to 5% or not. Furthermore, by inspecting Figures 4.15b and 4.27b, the results obtained on fold 1 show a greater mean predictive variability when utilizing fingerprints compared to not using fingerprints. In fact, this is consistent across all folds. Also, differences of both mean RMSEs and mean predictive variability demonstrate larger 95% approximate confidence intervals when utilizing fingerprints. Thus, adding chemical information seems to increase the variance of Macau. However, uncertainty sampling with fingerprints does not seem to find a training set different to the training set obtained by random sampling at an earlier step compared to uncertainty without fingerprints, which is illustrated by the comparison of the p-values in Figure 4.18 and 4.30. Figure 5.2 shows p-values from the Mann-Whitney U test between all training subsets (except for the initialization) of uncertainty sampling with fingerprints and uncertainty sampling without fingerprints obtained on fold 1 and 2 of the Merck data. Both cases did not utilize any correlation simulations. Since the majority of the p-values are greater than 0.05, utilizing fingerprints do not seem to generate a significant different training set, except for the initial iterations where the sets illustrate some differences.

Using fingerprints seems to result in a small improvement of the prediction accuracy of both uncertainty sampling and random sampling. However, using fingerprints seems to imply an increase in variance of the predictions. Also, using fingerprints does not seem to generate any different information about what data to add with respect to uncertainty without fingerprints, since a significantly different training set was not obtained when using fingerprints. A possible explanation of these results could be that Macau exploits the underlying experimental design rather than the chemical features. Thus, to provide reasonable predictions of the observed product yield, it seems to be enough to solve a combinatorial problem. Also, this could indicate that the fingerprints do not provide any further information that is essential for describing the differences between the choices of reaction components, or that the relationships in the experimental design is sufficient for the model to know where



(a) Fold 1 **(b)** Fold 2

Figure 5.2: The p-values of a Mann-Whitney U test of all subsets, except the initialization, of the training set with points from the Merck data when utilizing uncertainty sampling with and without fingerprints. No correlation simulations were utilized for both cases. p-values are not plotted for the subsets of the initialization. Only fold 1 and 2 are displayed but the other folds show similar behaviors.

it is uncertain. On the other hand, the increase in variance when using fingerprints could be an indication that the model becomes more uncertain about its predictions due to the additional information. This could indicate that the model improves its ability to express uncertainty in its prediction, but then we would expect an significant improvement in predictive variability of uncertainty sampling compared to random sampling, which have not consistently been observed over the different folds. Therefore, utilizing matrix factorization for synthesis predictions, information about the chemical structures is not needed, when finding relevant relationships in experimental design is of importance. However, when predicting a choice of reaction component without any training data, the additional chemical information that fingerprints provide should be important. Thus, additional chemical information should be required when performing synthesis predictions on a new domain, but this has not been investigated in this thesis.

5.2 Uncertainty in Neural Networks

This section will analyze the results obtained for the Bayes by Backprop and MC dropout approach in Section 4.3. Furthermore, the section will also discuss how these results should be interpreted in order for the networks to be useful decision making tools for synthesis prediction. We will start by comparing and analyzing the performance of the different models. Thereafter, the uncertainty of the models will be discussed. Note that the uncertainty only will be investigated on the dedicated test set, i.e., the set which consists of 10% of the observations.

5.2.1 Performance

The performance results of Bayes by Backprop in Section 4.2.1 illustrate that Bayes by Backprop is highly dependent on the value of β. This is to be expected since a lower value of β results in a less impactful complexity loss, i.e., the model is more trained to fit the data. A consequence of this is that the model is more prone to overfitting the data since the effect of the regularizing complexity loss is reduced. This is best illustrated by the likelihood loss of $\beta = 0.1$, Figure 4.31e, where the test and validation loss increases after the first few epochs, despite the monotonically decreasing training loss. Due to the overfitting illustrated by the $\beta = 0.1$ model, it was not investigated further.

For the investigated values on β, the value $\beta = 0.5$ illustrates the best predictive performance. This conclusion is explained by the fact that even though the model overfits the data, illustrated by lower training loss, the validation and test loss is also lower than the $\beta = 1.0$ model, as seen in Figure 4.31. That overfitting also reduces the validation and test loss indicates that there is some regularity in the data, i.e., the different splits must be similar. Furthermore, the top 1 accuracy of the $\beta = 0.5$ model is higher and the model manages to predict more ligands than the $\beta = 1.0$ model. The top 1 accuracy and predicted diversity are illustrated in Figures 4.32 and 4.33, respectively. It is worth noting that the model manages to find two or three more ligands with $\beta = 0.5$ compared to $\beta = 1.0$. However, the very scarce ligands, which account for approximately 2% of the data, were not predicted by any of these two models.

The loss of Bayes by Backprop in Figure 4.31 highlights an important drawback of the Bayes by Backprop approach; that the main contribution to the total loss, i.e., the sum of the likelihood and the complexity loss, can be the complexity loss. The problem is that the complexity loss is independent of the data, and only depends on the architecture of the network, the choice of prior and the assumed variational posterior. This means that this loss is not directly affected by the data used to train the model. Therefore, the accumulation of more data will not directly affect the complexity loss; however, it will greatly affect the likelihood loss, which is heavily dependent on the amount of available data. Thus, the relative effect of the complexity loss on the total loss decreases with the quantity of the data, since the impact of the likelihood loss increases. This is advantageous since it mitigates the contribution of the prior, reducing the importance of selecting the correct prior. Finally, the fact that the complexity loss is larger than the likelihood loss also indicates that the variational Gaussian posterior, and the Laplacian prior, might represent the true distribution poorly.

As previously mentioned, $\beta < 1$ reduces the contribution of the complexity loss, and thereby increases the effect of the data without the accumulation of more data. However, note that, for arbitrary values of $\beta \neq 1$, the complexity loss term does not have a theoretical foundation. This means that for an arbitrary β, the objective is not necessarily to minimize the Kullback-Leibler divergence between the variational and true posterior. However, the model still learns a distribution of each weight, which allows for Bayesian inference and uncertainty quantification, even though the distribution might not be the optimal w.r.t. the Kullback-Leibler divergence.

The results of the performance of MC dropout, Section 4.2.2 and in particular Figures 4.34 and 4.35, illustrate that the $p = 0.8$ model has a better predictive performance than the $p = 0.5$ model. That the $p = 0.5$ model has a similar top 1 accuracy as the $p = 0.8$ model but a larger negative log likelihood could indicate that the $p = 0.5$ model is less confident in its predictions. This means that the model guesses correctly, but that the softmax output on average is lower compared to the $p = 0.8$ model. A peculiar observation is that MC dropout is based upon dropout, which is a method that is used to reduce overfitting, and both MC dropout models still illustrate severe overfitting [39]. That both models overfit the data is clearly illustrated by the large discrepancy between the training loss and the validation/test loss, and by the fact that the validation/test loss increases after the first 50 epochs.

The top 1 accuracy for both MC dropout models (Figure 4.35) exceed the top 1 accuracy for Bayes by Backprop with $\beta \in \{1.0, 0.5\}$ (Figure 4.32). Furthermore, Figure 4.36 illustrates that both MC dropout models manage to predict at least 11 out of the possible 13 ligands, which is a significant improvement compared to the Bayes by Backprop models in Figure 4.33. However, the Bayes by Backprop models have a lower likelihood loss, on the validation and test split, compared to the MC dropout models, which is visualized in Figure 4.31 and 4.34, respectively. This could indicate that the Bayes by Backprop models are more certain about their correct classifications; or it could be a consequence of the overfitting by MC dropout, increasing the confidence in incorrect predictions.

A fundamental problem with the validation approach used to examine the performance of the Bayesian neural networks in this thesis is that it limits the model to predict already tested ligands of a reaction. This means that the model might predict a ligand that results in a valid reaction, however it will still be considered as incorrect if the reaction is yet to be tested, or simply absent from the current data. Thus, this validation approach limits the models to predict ligands that are favored by chemists, and it does not encourage exploration of previously unknown reaction conditions. If the model is very certain, but incorrect, on a given input then that could indicate that the prediction is a viable ligand for that reaction. However, the low predicted diversity for Bayes by Backprop does not provide enough evidence that the model performs at an adequate level, but rather indicates that the model struggles to learn the data.

5.2.2 Uncertainty

A key advantage of Bayesian models is that they can provide uncertainty estimates. This can be a valuable tool if the model is to be used for decision making since then it would be valuable to know how certain the model is, and if the prediction should be passed to a human for validation. However, it is essential that the uncertainty is interpretable, especially if decisions are executed based on the uncertainty.

The naïve uncertainty for 1000 forward passes in Figure 4.37 highlights that MC dropout in general is more certain and more inclined to predict the correct ligand, compared to Bayes by Backprop. This is illustrated by more large diagonal elements. This is consistent with the performance result for Bayes by Backprop, where

both models struggles to predict the minority ligands. We again emphasize that Figure 4.37 illustrates the spread of predictions for each true ligand. This means that a column containing multiple large elements does not necessarily indicate that the model is uncertain on a specific input since the predictions could originate from different inputs. However, the test set only contains a single observation for ligands 1, 4, 5, 6 and 7, thus these columns correspond to how uncertain the model is on that input, according to the naïve uncertainty.

The naïve uncertainty, Algorithm 5, counts how many times the model guesses each ligand. This means that the accumulated uncertainty is easy to interpret. However, a drawback of this method is that it disregards a lot of information in each forward pass. The reason for this is that only the element that corresponds to the largest output entry will be stored. Therefore, the information about how much larger the winning element of the output layer was, and how much that alters each forward pass, is ignored. This basically means that this uncertainty estimate can provide an output which indicates that the model is very certain, when in fact the winning element might only surpass the second largest element by a small margin each time. Bayes by Backprop with $\beta = 1.0$ illustrates this for the max entropy sample, Table 4.3, where the model guesses ligand 12 on 780 out of the 1000 forward passes. This seems to indicate that the model is very certain about the output, however, the variational predictive in Table 4.4 illustrates that the corresponding probability only is 0.26. Thereby, this uncertainty method could potentially overestimate how certain the model is. Finally, it is worth noting that this simple method to estimate uncertainty probably can be useful in combination with the variational predictive distribution. This is due to the fact that the method is easy to interpret and it provides some estimate of the variance in the output, and the variational predictive provides an estimate on how much more likely the output is.

Entropy was used as an informativeness measure to differentiate between uncertain and certain points. An advantage of entropy based uncertainty is that it is easy to compare between observations since the entropy is a scalar. However, the entropy of a given input does not provide much information about how certain the model is on the specific input. The exception is if the value is close to the minimal or maximal entropy, which is 0 and $\log(13)$, respectively. Note that 13 originates from the fact that there are 13 possible ligands in this data. Nonetheless, the metric could be useful if one compares the entropy between different observations since a larger value would indicate a more uncertain observation. A deficit of this method is that it does not provide information about the spread of the observation, since it is based on the average softmax. Furthermore, it does not include information about how likely each ligand is. Thus, this uncertainty is probably best to use in combination with the variational predictive distribution since this contains information about how likely each observation is. However, this does not address the issue about the absence of how the predictions alter between forward passes, i.e. the variance in the predictions, which will not be incorporated in this uncertainty estimate.

The decomposition into aleatoric and epistemic uncertainty returns the covariance matrix of the corresponding variational distribution. A consequence of this is that the uncertainty can be harder to interpret than the naïve uncertainty. However,

unlike the naïve uncertainty, this decomposition does not discard the information of the output distribution. Additionally, this method also encloses information about how the output varies between forward passes, which is not incorporated in the entropy uncertainty. Thereby, the obtained uncertainty from the decomposition could potentially be a more accurate description of the uncertainty in the model.

If the matrix contains multiple large elements then that indicates that the model is uncertain. This follows from the fact that then there are multiple entries that deviate significantly. Bayes by Backprop with $\beta = 1.0$ for the random input, Figure 4.38, and MC dropout for the "Max entropy" observation, Figure 4.41, are examples of this case. On the other hand, if the matrix contains only a single large element then the model could both be certain and uncertain. This interpretation can be explained by the fact that a large element could originate due to two reasons; one is that the model alters a lot between each forward pass, and that the variational predictive for the winning element is small, which indicates that the model is uncertain. The "Max entropy" observation in Figure 4.40 might correspond to such a case. This is since element $(8, 8)$ is much larger than the remaining elements, and the corresponding variational element is also rather low, Table 4.2, which could indicate that the model is uncertain. The other scenario is that the model indeed does alter; however, the deviations are only large because the winning element (largest element) in the variational predictive is large. That is, the absolute variations might be large while the relative variations are small, which indicates that the model is certain. This is illustrated by MC dropout in Figures 4.38e and 4.38f, where the total uncertainty of element $(1, 1)$ is approximately 0.2, however, the corresponding variational element is 0.72 as can be seen in Table 4.2. To conclude, if a single element is large then one should probably compare the value with the variational predictive to determine whether the model is certain or not. The final case that can arise is that the matrix does not contain any large elements, which simply implies that the model is certain on the given input. This is illustrated by, for example, the "Min entropy" observation for MC dropout, Figure 4.41. However, we emphasize that small uncertainties only implies that the model is consistent between runs, i.e., the output distribution does not alter significantly; it does not imply that the winning unit is greater by a margin. This means that while the output distribution of the model is consistent, the variational predictive might indicate that the winning unit only is slightly larger than other units.

A potential problem with the decomposition into aleatoric and epistemic uncertainty is that it assumes one-hot encoded vectors. While this assumption holds true for the true distribution, it might not be the case for the model distribution. This basically means that it might be advantageous if the output distribution contained multiple zeros and that the softmax layer might converge too slowly to such a distribution. An important direction for future work could thus be to investigate if there exists a more suitable output distribution, or if softmax converges fast enough. The decomposition is also based on variational inference, which assumes that the true posterior can be replaced by a variational distribution. However, the approximating distributions might not be sufficiently representative of the true distributions, which thereby induces a lot of errors in the decomposition. In particular, the variational predictive

$q(\boldsymbol{y}^*|\boldsymbol{x}^*, \theta)$ might not be representative of the true posterior predictive distribution $\pi(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D})$, i.e., the samples generated from the variational predictive might not be adequate.

A slightly worrying observation is that the aleatoric uncertainty in general is larger than the epistemic uncertainty, e.g., Figure 4.39. This is worrying since what the aleatoric uncertainty represents for fingerprints is unknown. It might just relate to input errors when chemists translate their reaction notes to fingerprints, however the fact that the aleatoric uncertainty seems to be larger than the epistemic uncertainty for most observations implies that this explanation is unlikely. That the aleatoric part contains a term which only depends on the model likelihood, and not the model likelihood squared, could potentially be the origin of the large aleatoric uncertainty. Therefore, if the assumption that the model outputs a one-hot encoded vector is violated, then that will probably impact the squared terms more than the non squared term. Thus, it is of great interest to investigate how the uncertainty, and especially the aleatoric uncertainty, alters if a different model likelihood is used, i.e., a different output layer than the softmax layer.

The epistemic uncertainty measures the expected variance from the expected output of the model. Thus, a low epistemic uncertainty corresponds to a model that, on average, does not deviate too much from the mean prediction of the model. This is what we interpret as a certain model. The fact that the epistemic uncertainty is significantly lower for observations with low entropy, compared to high entropy, indicates that a low epistemic uncertainty corresponds to a model where the output does not depend significantly on the weight sample, i.e., the model is certain.

The results illustrate that uncertainty can be induced to neural networks, i.e., the networks can provide an estimate of how reliable the output is. The uncertainty can potentially be very useful if the model is to be used as a tool for decision making. In particular, for chemical synthesis, the model can provide a suggestion of what ligand to use in a reaction, and it can also provide an estimate of how certain the model is, which can guide the chemist performing the synthesis experiment for obtaining new hit compounds. The fact that the models achieve a top 1 accuracy of around 80% on both the test and validation set indicates that the most frequent prediction of the forward passes can be used as a suggestion on what ligand to use in a reaction. Furthermore, if the three most frequent predictions are used then the models seem to be correct approximately 95% of the time. In synthesis planing, this means that the chemist can be quite confident that one of the three most probable suggestions will be a viable candidate for the reaction.

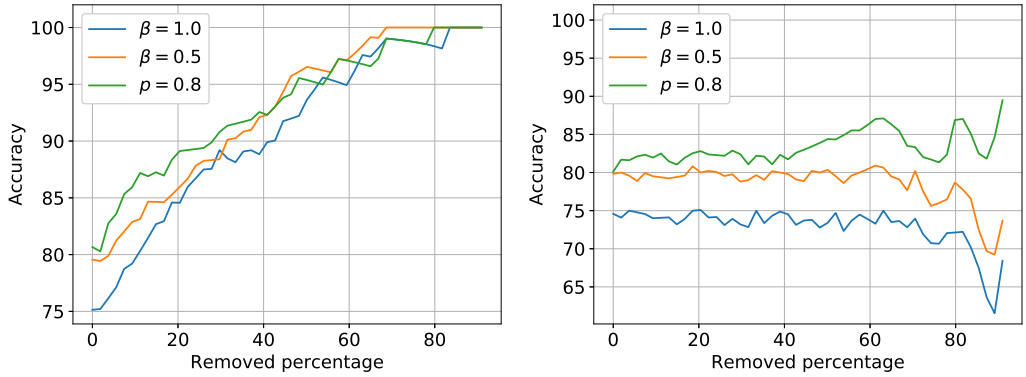
An example of how the uncertainty can guide the decision making is apparent in the naïve uncertainty of the "Max entropy" observation for MC dropout, Table 4.7. If the suggested ligand of the network is the most frequent of the 1000 forward passes then the network would suggest ligand 11, which in this case also is correct. However, this single prediction does not contain the uncertainty of the network. A consequence of this is that the chemist utilizing the model has no information about the model's confidence in the output, and thereby naïvely uses the predicted ligand. However, from the uncertainty estimate it is obvious that the model is uncertain on which ligand to use in the reaction. Therefore, it can be concluded that in this case

the chemist should not base the synthesis experiment on the network's prediction but rather on his/her expertise. Additionally, the uncertain prediction could also indicate that more data is needed in order for the model to perform better. On the other hand, if the network is certain on the input, e.g., the "Min entropy" observation for MC dropout in Table 4.7, then the network indicates that the most frequently predicted ligand could be used in the experiment. That the model can distinguish between certain and uncertain predictions is very useful if the model is to be used in a semi-autonomous lab, since certain predictions potentially could be used naïvely and uncertain predictions could be handled by an external source.

However, in order for the uncertainty to be truly useful in decision making it is important to investigate if the models are well-calibrated, i.e., is the uncertainty in the model actually correlated with how the model performs. This means that an input where the model is certain should ideally correspond to a correct classification. Worst case is if the model is certain and incorrect, which is the case for the input corresponding to ligand 4, where all models incorrectly predict ligand 9 more than 90% of the time, Figure 4.37. The naïve uncertainty in Figure 4.37 illustrates that both Bayes by Backprop models are uncertain about the input corresponding to ligand 5 since the column contains multiple large elements; however, neither of the models predict ligand 5. MC dropout, on the other hand, indicates that it is rather certain on the input, but it incorrectly classifies the input as ligand 9 instead of ligand 5, even though ligand 5 also is predicted. This highlights an important aspect of the induced uncertainty: that a certain output does not necessarily correspond to a correct prediction. A potential problem with this is that if the model is used for synthesis planning, then the utilizers of the model might be overly confident in the network's prediction, especially if the uncertainty is low. This basically means that a certain model can be mistakenly interpreted as a correct model. In an autonomous or automatic setting for synthesizing new hit compounds, this confusion could potentially be both expensive and time-consuming since the model might suggest multiple faulty reactions with low uncertainty. This further emphasizes the importance to investigate the quality of the uncertainty.

Figure 5.3 illustrates some preliminary results for the quality of the entropy and total uncertainty estimate, i.e., whether the model is well-calibrated or not. The results were obtained by successively removing the data points corresponding to the maximum total uncertainty or entropy from the test set. Figure 5.3a visualizes that removing data points with a large variational entropy indeed yields a better accuracy. This implies that points with high entropy seem to correspond to points for which the model is incorrect and uncertain. The results in Figure 5.3b indicates that simply ranking points according to their maximum total uncertainty, i.e., the sum of aleatoric and epistemic, does not yield an increase in predictive performance. However, this is consistent with the interpretation of the uncertainty matrix, which was described previously in this section. Therefore, an important area for future research is to investigate how the decomposed uncertainty can be utilized, or if the current simple utilization is sufficient, which in that case implies that the decomposition obtained from these models are inferior. Note that this evaluation method does not take into account whether the prediction is correct or not, since it removes points

solely based on the uncertainty. That is, an uncertain and correct model is equally bad as an uncertain and incorrect model. However, these results are included merely to highlight the importance of investigating the quality of the obtained uncertainty.



(a) Entropy **(b)** Total uncertainty

Figure 5.3: How the test accuracy alters when the most uncertain points, according to entropy or the total uncertainty (sum of aleatoric and epistemic), are removed for Bayes by Backprop with $\beta \in \{1.0, 0.5\}$ and MC dropout with $p = 0.8$. Note that removing the points with the largest entropy generally increases the accuracy. Removing the points with the largest total uncertainty does not illustrate such a trend.

5.3 Future Work

5.3.1 Active Learning Using Uncertainty Quantification

To develop a less greedy active learning algorithm using uncertainty is believed to be a reasonable step in the right direction. This project tried to implement a less greedy algorithm by simulating and using correlations between points. However, this was done without any success. Therefore, other ways to determine correlations between points could be interesting to explore and evaluate in order to see if they have any significant effect. Furthermore, our approach utilizes the predictive uncertainty, which is the combined aleatoric and epistemic uncertainty. Separating the aleatoric and epistemic uncertainty can enable the use of a less greedy approach of using the uncertainty.

When utilizing uncertainty in active learning the goal was to lower the variance of predictions by adding data corresponding to predictions that the model is most uncertain about. As expected, we have observed that the predictive variability, i.e., the overall variance of the predictions, decreases when the number of points in the training set is increased. Also, the RMSE seems to decrease in the same way. It would therefore be interesting to compare the absolute error and variance of each prediction in order to investigate if the points with low variance also are the points with low absolute error. That is, explore how the error and variance of the prediction

is related. It is often desired to obtain both a low error and a low uncertainty from a model, since this can give further indications on whether this kind of approach is reasonable or not, or if another approach should be applied. Also, this would further investigate the quality of the uncertainty, which is important for future work.

To validate the results and further explore the use of uncertainty in active learning, a larger data set needs to be investigated. For instance, it would be interesting to explore a more complex experimental design to study how it affects the performance and how the use of additional information of chemical features would affect the uncertainty quantification. Also, it would be interesting to investigate AL for exploring data which no training data covers, so called out-of-matrix predictions. This would be important when using synthesis prediction to explore new domains of the chemical space. Moreover, in order to examine the effect of fingerprints and if it provides any essential information, fingerprints with different numbers of bits should be investigated in future studies.

5.3.2 Uncertainty in Neural Networks

The large complexity loss obtained by the Bayes by Backprop models indicate that the choice of prior and posterior can be very influential. Thus, in order to use Bayes by Backprop successfully one should further investigate how to select an appropriate prior and posterior. Furthermore, the parameter β has a large impact on the model performance, so to optimize the model performance w.r.t. β could be of great interest.

The advantage of utilizing Bayesian neural networks is that they can provide an estimate of how certain a model is. This can be very useful if the model is to be used as a tool for decision making. In particular, for chemical synthesis the model can provide a suggestion of what ligand to use in a reaction, and it can also provide an estimate of how certain the model is in the suggestion. However, the usefulness of the uncertainty measure depends on whether the models are well-calibrated or not. Therefore, it is interesting to investigate the quality of the uncertainty obtained from the different uncertainty estimation methods. The preliminary results of the quality of the estimated uncertainty indicate that a large entropy seems to correspond to more uncertain observations, Figure 5.3a. The simple quality estimation of aleatoric and epistemic uncertainty, as seen in Figure 5.3b, did not imply that a large total uncertainty corresponds to an uncertain prediction. This is consistent with the interpretation of the uncertainty matrix. Therefore, it remains to investigate how the quality of the aleatoric and epistemic uncertainty can be deduced and how to use this uncertainty; or if the preliminary results simply implies that these estimates are not correlated with better performance.

If the models, indeed, are well-calibrated then there are numerous potential applications of the uncertainty from Bayesian neural networks, besides merely a tool for decision making. One possible application could be to investigate if the uncertainty can be feedback during training in order to increase model performance. The idea is that if the model is certain and incorrect, then the model should be punished harder compared to if the model is uncertain and incorrect. Another application would

be to use the uncertainties in active learning. The results obtained by the matrix factorization method indicate that the use of uncertainties in active learning has an effect on the predictive performance of a model. However, this thesis did not investigate the use of uncertainty-based active learning for Bayesian neural networks, and therefore this is an area for future work. A remark to the active learning application is that if the model is well-calibrated and it is very certain on an input which it classifies incorrectly, then that could provide evidence that the reaction should be tested again. That is, the model can be used to suggest which future experiments to conduct, which potentially could reduce the time required to synthesize possible hit compounds. The active learning application is also an essential component in an autonomous lab.

Lastly, the current implementation does not take yield of the reaction into consideration. This means that the model only possesses the binary information on whether a reaction is possible or not. However, to incorporate the associated yield into the model can be very useful since the model then might be able to differentiate between optimal and suboptimal conditions. This could further enhance the usefulness of the model, especially if it is to be utilized to suggest which experiments to conduct.

6

Conclusion

6.1 Active Learning Using Uncertainty

The use of uncertainty for active learning demonstrates a slightly better performance compared to selecting points randomly. However, the benefit of uncertainty is apparent when a sufficient number of points have been added to the training set which, therefore, does not necessarily yield a reduction of the time required to find hit compounds unless enough experiments are performed. Moreover, the training sets obtained by utilizing uncertainty and selecting random points are shown to be significantly different; while both strategies show decreased absolute error and variance when increasing the number of points in the training set. Thus, the state-of-the-art model is able to exploit patterns within the underlying experimental design without using any information about the uncertainty in its predictions, and as a consequence the discrepancy in performance with and without uncertainty is small.

6.2 Uncertainty in Neural Networks

For Bayes by Backprop the hyperparameter β greatly affects the performance of the model. In particular, for $\beta < 1$ the model is more encouraged to fit the data. $\beta = 0.5$ illustrated the best predictive performance, and it was comparable to the performance of MC dropout with $p = 0.8$. In particular, both models manages to consistently achieve a top 1 accuracy of at least 80%. This indicates that the top 1 prediction of the models probably can be used to predict which ligand to use in a synthesis experiment.

The added feature of uncertainty quantification of the Bayesian networks can become an useful tool for decision making in chemical synthesis experiments. The naïve uncertainty is easy to interpret but it discards a lot of the information of the model's output. The decomposition into aleatoric and epistemic uncertainty, on the other hand, contains more information about how the output from the model varies. However, this uncertainty is harder to interpret and more difficult to use in an autonomous setting. Note that all of the tested uncertainties preferably should be used in combination with the probabilities of the variational predictive distribution. Lastly, the preliminary results of the quality in Figure 5.3, indicates that entropy could be an adequate uncertainty measure; however, further research of this is needed.

6.3 Uncertainty Quantification in Synthesis Prediction Models

The following general conclusions of uncertainty quantification in chemical prediction models are made:

- The uncertainty can be used as information about where a synthesis prediction model can be improved when a sufficient amount of data is available.
- Uncertainty in the predictions can provide valuable information for the decision making process in synthesis prediction. However, it is critical to understand how the uncertainty should be utilized in order to make reliable decisions.
- To use uncertainty it is essential to further investigate its quality, i.e., what does uncertain predictions represent and how is the quality established/guaranteed?

Bibliography

[1] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, "The rise of deep learning in drug discovery," *Drug Discovery Today*, vol. 23, no. 6, pp. 1241–1250, Jun. 2018, doi:10.1016/j.drudis.2018.01.039.

[2] A. F. de Almeida, R. Moreira, and T. Rodrigues, "Synthetic organic chemistry driven by artificial intelligence," *Nature Reviews Chemistry*, vol. 3, no. 10, pp. 589–604, Aug. 2019, doi:10.1038/s41570-019-0124-0.

[3] C. W. Coley, N. S. Eyke, and K. F. Jensen, "Autonomous discovery in the chemical sciences part ii: Outlook," *Angewandte Chemie International Edition*, Sep. 2019, doi:10.1002/anie.201909989.

[4] P. Raboisson, D. Rognan, D. Aldous, and C. G. Wermuth, *Practice of Medicinal Chemistry*, 4th ed. London, UK: Elsevier, 2015. [Online]. Available: https://www.sciencedirect.com/book/9780124172050, Accessed on: Apr. 6, 2020.

[5] J. A. DiMasi, H. G. Grabowski, and R. W. Hansen, "Innovation in the pharmaceutical industry: New estimates of R&D costs," *Journal of Health Economics*, vol. 47, pp. 20–33, May 2016, doi:10.1016/j.jhealeco.2016.01.012.

[6] E. N. Feinberg *et al.*, "Potentialnet for molecular property prediction," *ACS Central Science*, vol. 4, no. 11, pp. 1520–1530, Nov. 2018, doi:10.1021/acscentsci.8b00507.

[7] K. Yang *et al.*, "Analyzing learned molecular representations for property prediction," *Journal of Chemical Information and Modeling*, vol. 59, no. 8, pp. 3370–3388, Jul. 2019, doi:10.1021/acs.jcim.9b00237.

[8] L. Wang, J. Ding, L. Pan, D. Cao, H. Jiang, and X. Ding, "Artificial intelligence facilitates drug design in the big data era," *Chemometrics and Intelligent Laboratory Systems*, vol. 194, Nov. 2019, doi:10.1016/j.chemolab.2019.103850.

[9] M. Popova, O. Isayev, and A. Tropsha, "Deep reinforcement learning for de novo drug design," *Science Advances*, vol. 4, no. 7, p. eaap7885, Jul. 2018, doi:10.1126/sciadv.aap7885.

[10] P. Schwaller *et al.*, "Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction," *ACS Central Science*, vol. 5, no. 9, pp. 1572–1583, Aug. 2019, doi:10.1021/acscentsci.9b00576.

[11] C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, and K. F. Jensen,

"Prediction of organic reaction outcomes using machine learning," *ACS Central Science*, vol. 3, no. 5, pp. 434–443, Apr. 2017, doi:10.1021/acscentsci.7b00064.

[12] C. W. Coley, W. H. Green, and K. F. Jensen, "Machine learning in computer-aided synthesis planning," *Accounts of Chemical Research*, vol. 51, no. 5, pp. 1281–1289, May 2018, doi:10.1021/acs.accounts.8b00087.

[13] F. Peiretti and J. M. Brunel, "Artificial intelligence: the future for organic chemistry?" *ACS Omega*, vol. 3, no. 10, pp. 13 263–13 266, Oct. 2018, doi:10.1021/acsomega.8b01773.

[14] R. J. Ouellette and J. D. Rawn, *Organic chemistry: structure, mechanism, and synthesis*. San Diego, CA, USA: Elsevier, 2014. [Online]. Available: https://www.sciencedirect.com/book/9780128007808/organic-chemistry, Accessed on: Apr. 27, 2020.

[15] G. Schneider, "Automating drug discovery," *Nature Reviews Drug Discovery*, vol. 17, no. 2, pp. 97–113, Feb. 2018, doi:10.1038/nrd.2017.232.

[16] E. Begoli, T. Bhattacharya, and D. Kusnezov, "The need for uncertainty quantification in machine-assisted medical decision making," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 20–23, Jan. 2019, doi:10.1038/s42256-018-0004-1.

[17] Y. Zhang and A. A. Lee, "Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning," *Chemical Science*, vol. 10, no. 35, pp. 8154–8163, Jul. 2019, doi:10.1039/C9SC00616H.

[18] M. K. Warmuth, J. Liao, G. Rätsch, M. Mathieson, S. Putta, and C. Lemmen, "Active learning with support vector machines in the drug discovery process," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 2, pp. 667–673, Feb. 2003, doi:10.1021/ci025620t.

[19] R. S. Bohacek, C. McMartin, and W. C. Guida, "The art and practice of structure-based drug design: a molecular modeling perspective," *Medicinal Research Reviews*, vol. 16, no. 1, pp. 3–50, Jan. 1996, doi:10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6.

[20] D. Perera *et al.*, "A platform for automated nanomole-scale reaction screening and micromole-scale synthesis in flow," *Science*, vol. 359, no. 6374, pp. 429–434, Jan. 2018, doi:10.1126/science.aap9112.

[21] N. Miyaura and A. Suzuki, "Palladium-catalyzed cross-coupling reactions of organoboron compounds," *Chemical reviews*, vol. 95, no. 7, pp. 2457–2483, Nov. 1995, doi:10.1021/cr00039a007.

[22] D. T. Ahneman, J. G. Estrada, S. Lin, S. D. Dreher, and A. G. Doyle, "Predicting reaction performance in C–N cross-coupling using machine learning," *Science*, vol. 360, no. 6385, pp. 186–190, Apr. 2018, doi:10.1126/science.aar5169.

[23] K. V. Chuang and M. J. Keiser, "Comment on "Predicting reaction performance in C–N cross-coupling using machine learning"," *Science*, vol. 362, no. 6416, p. eaat8603, Nov. 2018, doi:10.1126/science.aat8603.

[24] J. G. Estrada, D. T. Ahneman, R. P. Sheridan, S. D. Dreher, and A. G. Doyle, "Response to Comment on "Predicting reaction performance in C–N cross-

coupling using machine learning"," *Science*, vol. 362, no. 6416, p. eaat8763, Nov. 2018, doi:10.1126/science.aat8763.

[25] C. M. Bishop, *Pattern recognition and machine learning*. New York, NY, USA: Springer, 2006. [Online]. Available: https://www.springer.com/gp/book/9780387310732, Accessed on: Mar. 12, 2020.

[26] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Department of Engineering, University of Cambridge, Cambridge, UK, 2016. [Online]. Available: http://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf, Accessed on: Feb. 10, 2020.

[27] J. A. Rice, *Mathematical statistics and data analysis*, 3rd ed. Belmont, CA, USA: Duxbury, 2007.

[28] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning : data mining, inference, and prediction*, 2nd ed. New York, NY, USA: Springer, 2009.

[29] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017, doi:10.1080/01621459.2017.1285773.

[30] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, May 2015. [Online]. Available: https://arxiv.org/pdf/1505.05424.pdf, Accessed on: Feb. 10, 2020.

[31] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5580–5590. [Online]. Available: https://dl.acm.org/doi/pdf/10.5555/3295222.3295309, Accessed on: Mar. 2, 2020.

[32] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? Does it matter?" *Structural Safety*, vol. 31, no. 2, pp. 105–112, Mar. 2009, doi:10.1016/j.strusafe.2008.06.020.

[33] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis*, vol. 142, p. 106816, Feb. 2020, doi:10.1016/j.csda.2019.106816.

[34] M. A. Nielsen, *Neural networks and deep learning*. San Francisco, CA, USA: Determination Press, 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/, Accessed on: Mar. 23, 2020.

[35] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991, doi:10.1016/0893-6080(91)90009-T.

[36] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989, doi:10.1007/BF02551274.

[37] B. Mehlig, "Artificial neural networks," *arXiv preprint arXiv:1901.05639*, Feb. 2019. [Online]. Available: https://arxiv.org/pdf/1901.05639.pdf, Accessed on: Mar. 11, 2020.

[38] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf, Accessed on: Mar. 17, 2020.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, Jan. 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html, Accessed on: Mar. 15, 2020.

[40] A. Graves, "Practical variational inference for neural networks," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, 2011, p. 2348–2356. [Online]. Available: https://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf, Accessed on: Feb. 12, 2020.

[41] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, May 2014. [Online]. Available: https://arxiv.org/pdf/1312.6114.pdf, Accessed on: Feb. 15, 2020.

[42] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning (ICML)*, 2016, pp. 1050–1059. [Online]. Available: http://www.jmlr.org/proceedings/papers/v48/gal16.pdf, Accessed on: Mar. 12, 2020.

[43] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009, doi:10.1109/MC.2009.263.

[44] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 720–727. [Online]. Available: https://www.aaai.org/Papers/ICML/2003/ICML03-094.pdf, Accessed on: Mar. 2, 2020.

[45] J. Simm *et al.*, "Macau: scalable Bayesian multi-relational factorization with side information using MCMC," *arXiv preprint arXiv:1509.04610*, Dec. 2015. [Online]. Available: https://arxiv.org/pdf/1509.04610.pdf, Accessed on: Feb. 18, 2020.

[46] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep. 1936, doi:10.1007/BF02288367.

[47] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *NIPS'07: Proceedings of the 20th International Conference on Neural Information Processing Systems*, 2007, pp. 1257–1264. [Online]. Available: http:

//papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf, Accessed on: Mar. 3, 2020.

[48] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *ICML'08: Proceedings of the 25th international conference on Machine learning*, 2008, pp. 880–887. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/1390156.1390267, Accessed on: Mar. 4, 2020.

[49] R. W. Kennard and L. A. Stone, "Computer aided design of experiments," *Technometrics*, vol. 11, no. 1, pp. 137–148, Feb. 1969, doi:10.1080/00401706.1969.10490666.

[50] N. M. O'Boyle and R. A. Sayle, "Comparing structural fingerprints using a literature-based similarity benchmark," *Journal of Cheminformatics*, vol. 8, no. 1, pp. 36:1–36:14, Jul. 2016, doi:10.1186/s13321-016-0148-0.

[51] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, Apr. 2010, doi:10.1021/ci100050t.

[52] H. L. Morgan, "The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service." *Journal of Chemical Documentation*, vol. 5, no. 2, pp. 107–113, May 1965, doi:10.1021/c160017a018.

[53] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, vol. 18, no. 1, pp. 50–60, Mar. 1947, doi:0.1214/aoms/1177730491.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, Jan. 2017. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf, Accessed on: Apr. 9, 2020.

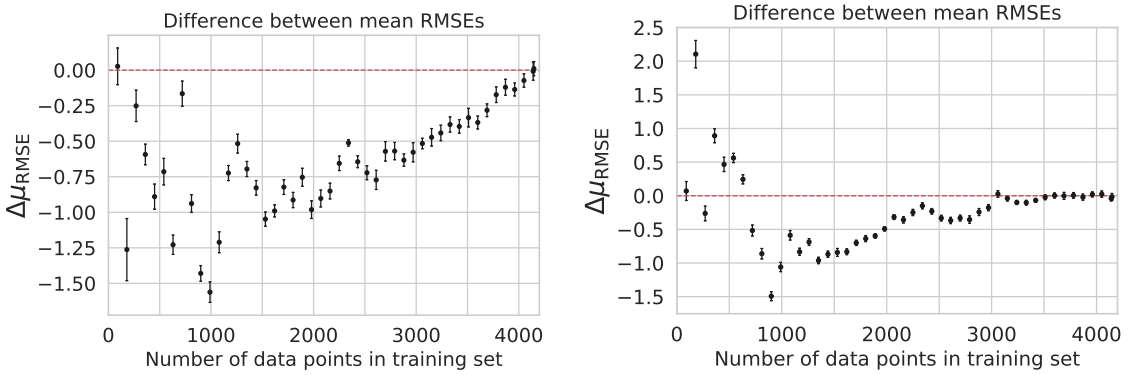
A

Additional Results

A.1 Active Learning Using Uncertainty

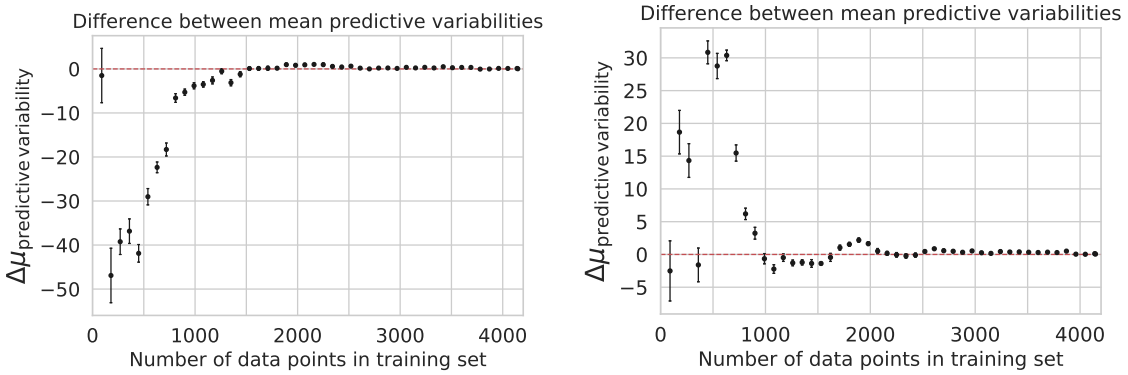
A.1.1 (Absolute) Uncertainty Compared to Relative Uncertainty

A.1.1.1 Pfizer Data



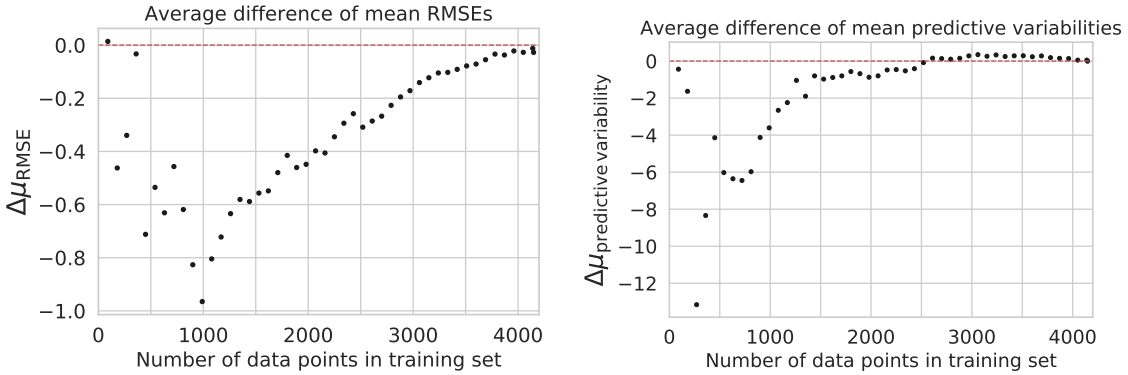
(a) Fold 1 **(b)** Fold 2

Figure A.1: The difference $\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{relative uncertainty}}$ of the means of the RMSE between (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations as a function of the number of points from the Pfizer data that have been added to the training set. The means was calculated from ten predictions runs. Random initialization was utilized, no correlation simulation was applied. Fold 1 and 2, of in total ten splits, of the cross validation is displayed. The red dotted line displays where the y-axis is equal to zero and the errors bars show the 95% approximate confidence interval.



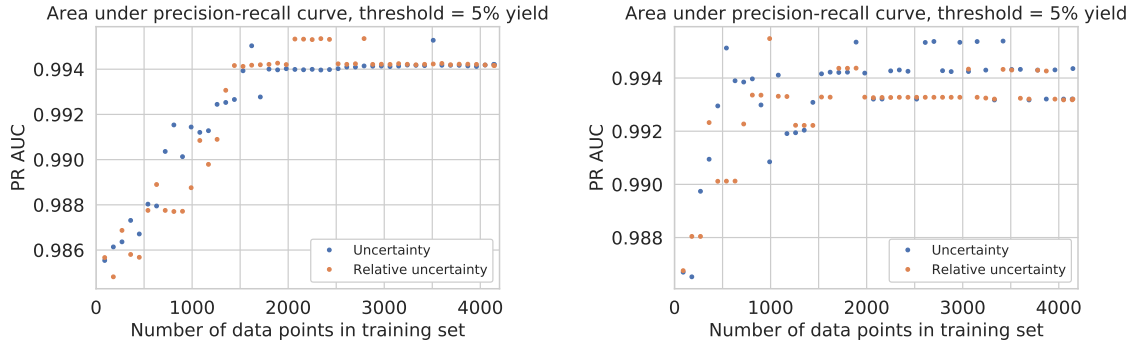
(a) Fold 1 **(b)** Fold 2

Figure A.2: The difference $\Delta\mu_{\text{predictive variability}} = \mu^{\text{uncertainty}}_{\text{predictive variability}} - \mu^{\text{relative uncertainty}}_{\text{predictive variability}}$ of mean predictive variability of (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations as a function of the number of points from the Pfizer data that have been added to the training set. Fold 1 and 2, of in total ten splits, of the cross validation is displayed. The red dotted line displays where the y-axis is equal to zero and the errors bars show the 95% approximate confidence interval.



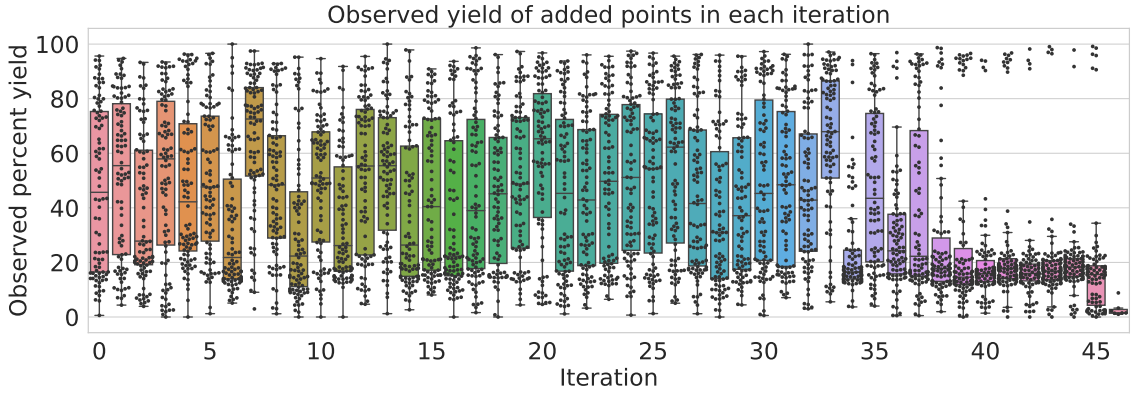
(a) RMSE **(b)** Predictive variability

Figure A.3: The average differences $\Delta\mu_{\text{predictive variability}}$ and $\Delta\mu_{\text{RMSE}}$ of (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations over all splits as a function of the number of points from the Pfizer data that have been added to the training set. The red dotted line displays where the y-axis is equal to zero and the errors bars show the 95% approximate confidence interval.

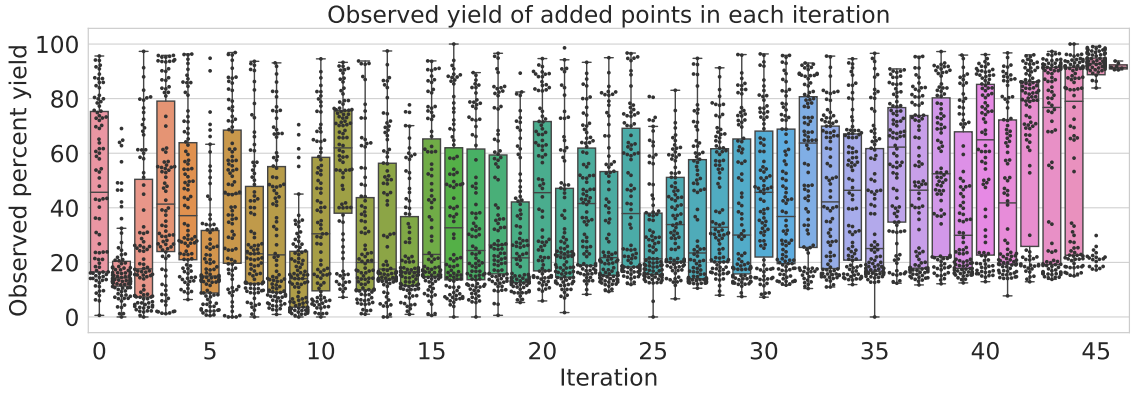


(a) Fold 1 **(b)** Fold 2

Figure A.4: Area under precision-recall curve of (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations applied on the Pfizer data.



(a) (Absolute) uncertainty sampling

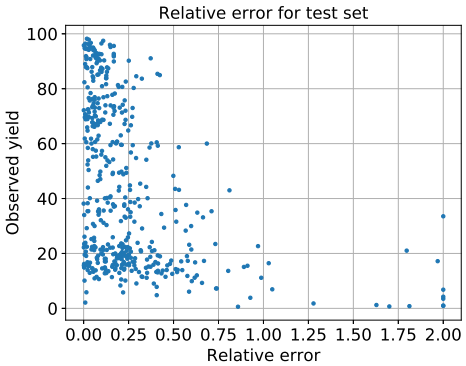


(b) Relative uncertainty sampling

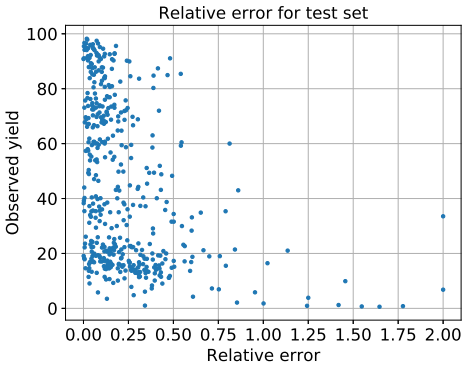
Figure A.5: Observed percent yield of points from the Pfizer data that have been added to the training set in each iteration when utilizing (absolute) uncertainty sampling without correlation simulations and relative uncertainty sampling without correlation simulations. Only split 1 is displayed but the other splits show similar behaviors.

(a) Fold 1: (Absolute) uncertainty sampling

(b) Fold 1: Relative uncertainty sampling



(c) Fold 2: (Absolute) uncertainty sampling

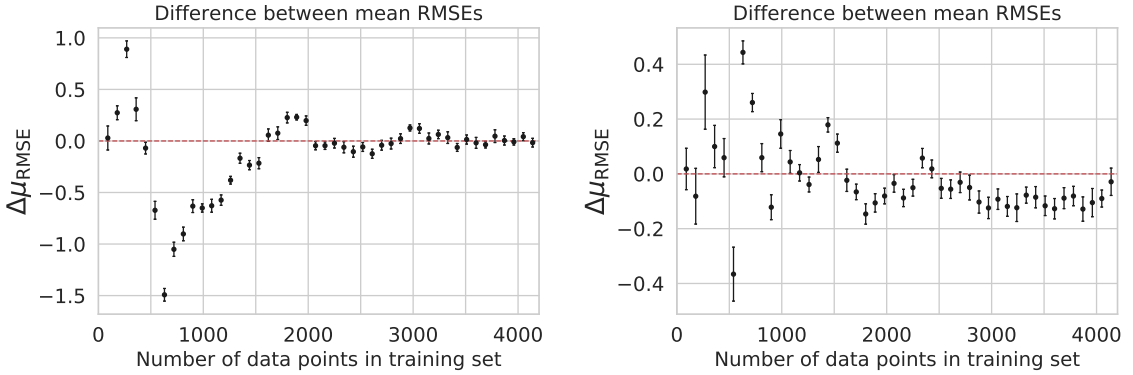


(d) Fold 2: Relative uncertainty sampling

Figure A.6: Observed yield plotted against relative error of the test set when utilizing (absolute) uncertainty sampling without correlation simulations or relative uncertainty sampling without correlation simulations applied on the Pfizer data. Only split 1 and 2 is displayed but the other splits show similar behaviors. Macau was trained by using points that have been added to the training set after the 16th iterations of the different sampling strategies.

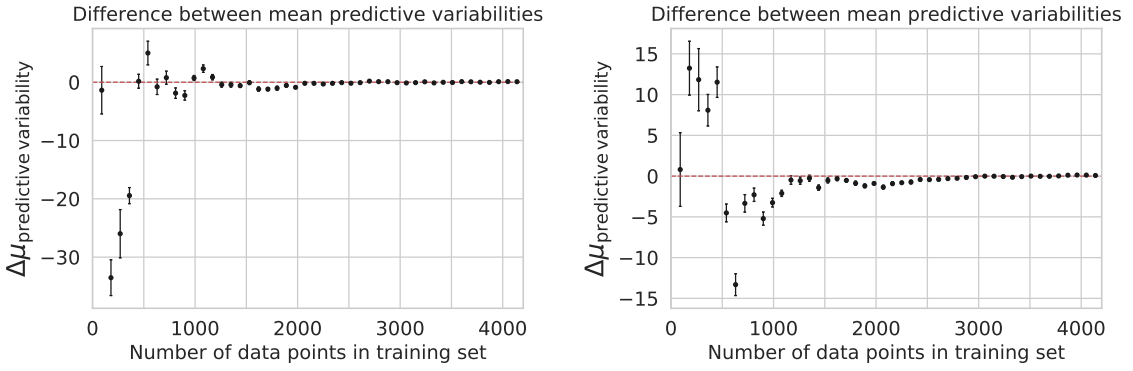
A.1.2 With and Without Correlation Simulations

A.1.2.1 Merck Data



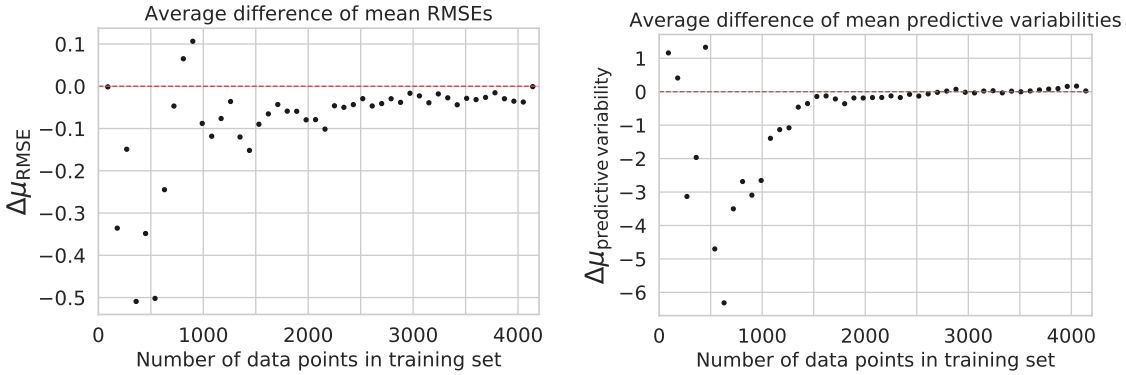
(a) Fold 1 **(b)** Fold 2

Figure A.7: The difference $\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{uncertainty without correlation}}$ of the mean RMSEs between uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations as a function of the number of points from the Merck data that have been added to the training set. Fold 1 and 2, of in total ten splits, of the cross validation are displayed. The red dotted line displays where the y-axis is equal to zero and the errors bars show the 95% approximate confidence interval.



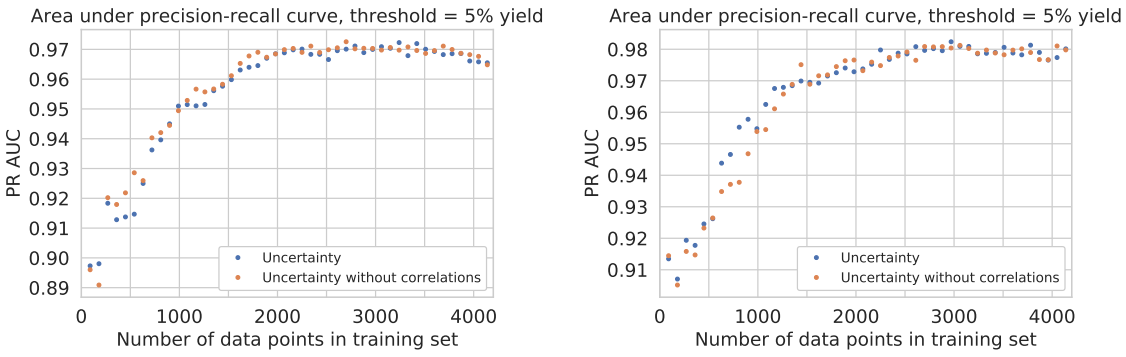
(a) Fold 1 **(b)** Fold 2

Figure A.8: The difference $\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty}} - \mu_{\text{predictive variability}}^{\text{uncertainty without correlation}}$ of the mean predictive variabilities between uncertainty sampling and uncertainty sampling without correlation simulations as a function of the number of points from the Merck data that have been added to the training set. Fold 1 and 2, of in total ten splits, of the cross validation are displayed. The red dotted line displays where the y-axis is equal to zero and the errors bars show the 95% approximate confidence interval.



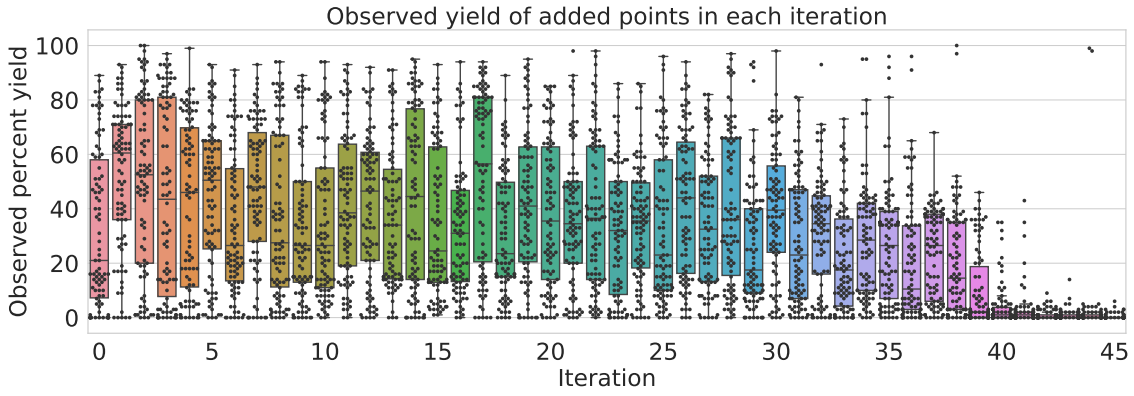
(a) RMSE **(b)** Predictive variability

Figure A.9: The average differences $\Delta\mu_{\text{predictive variability}}$ and $\Delta\mu_{\text{RMSE}}$ of uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulation over all splits as a function of the number of points from the Merck data that have been added to the training set. The red dotted line displays where the y-axis is equal to zero.

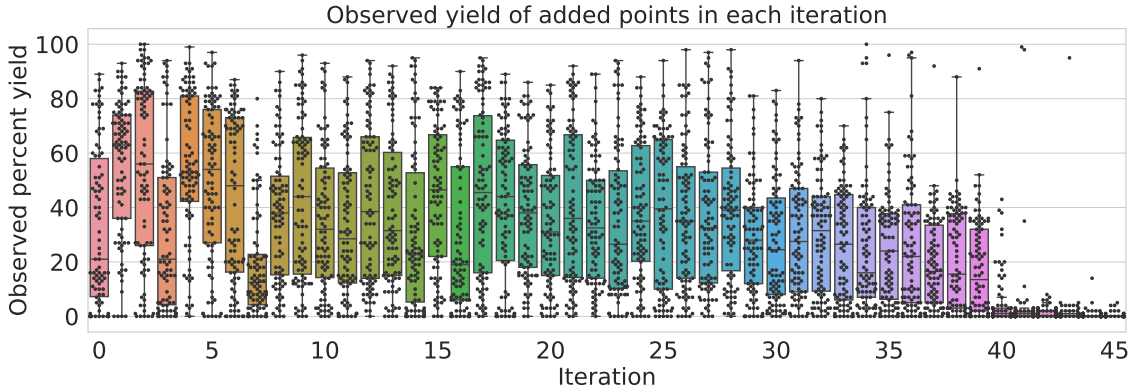


(a) Fold 1 **(b)** Fold 2

Figure A.10: Area under precision-recall curve of uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations applied on the Merck data.



(a) Uncertainty sampling without correlation simulation

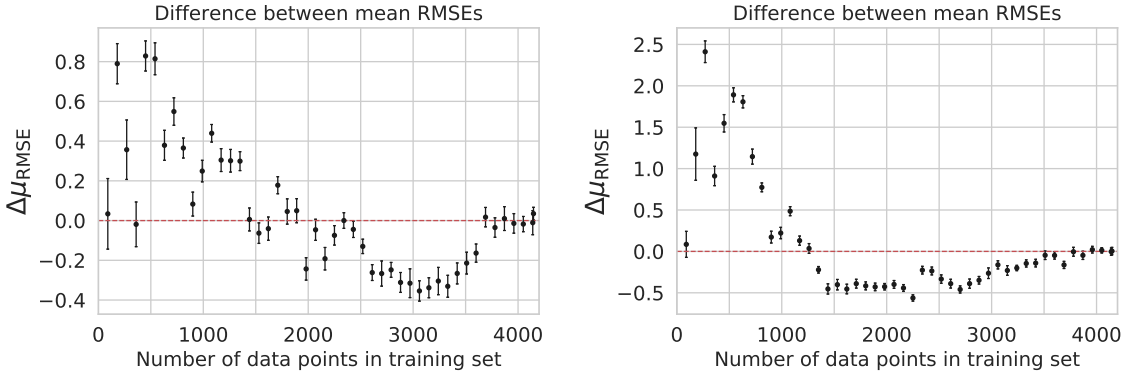


(b) Uncertainty sampling

Figure A.11: Observed percent yield of points from the Merck data that have been added to the training set in each iteration when utilizing uncertainty sampling with correlation simulations and uncertainty sampling without correlation simulations. Only split 1 is displayed but the other splits show similar behaviors. Iteration 0 is the initialization, which was done by random sampling.

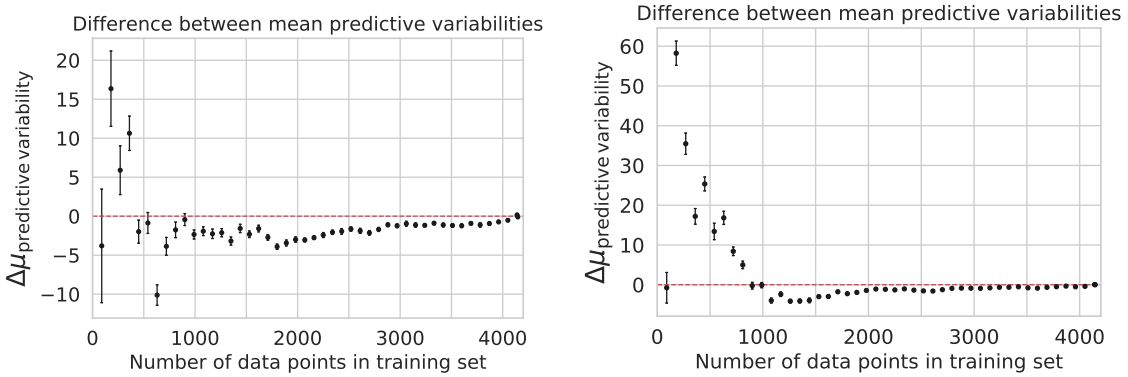
A.1.3 Uncertainty Compared to Random

A.1.3.1 Pfizer Data



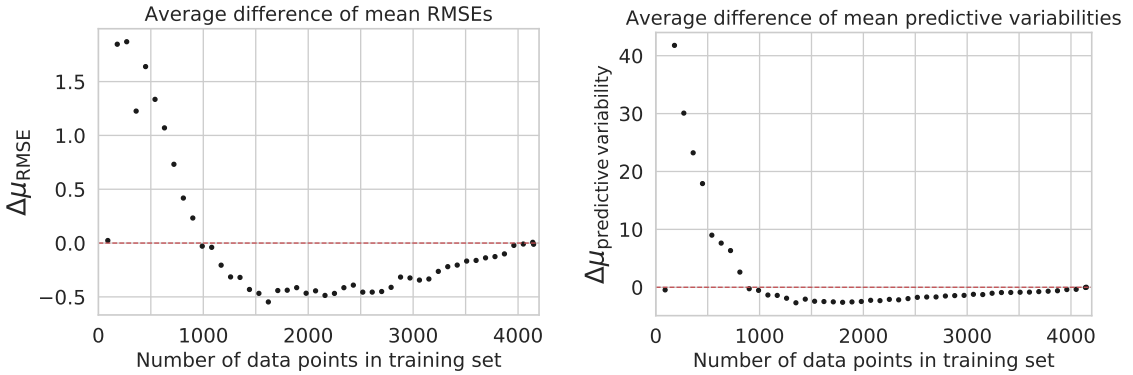
(a) Fold 1 **(b)** Fold 2

Figure A.12: The difference $\Delta\mu_{\text{RMSE}} = \mu_{\text{RMSE}}^{\text{uncertainty}} - \mu_{\text{RMSE}}^{\text{random}}$ of the mean RMSEs between uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Pfizer data that have been added to the training set. Fold 1 and 2, of in total ten splits, of the cross validation are displayed. The red dotted line displays where the difference is equal to zero and the error bars display the approximate 95% confidence intervals.



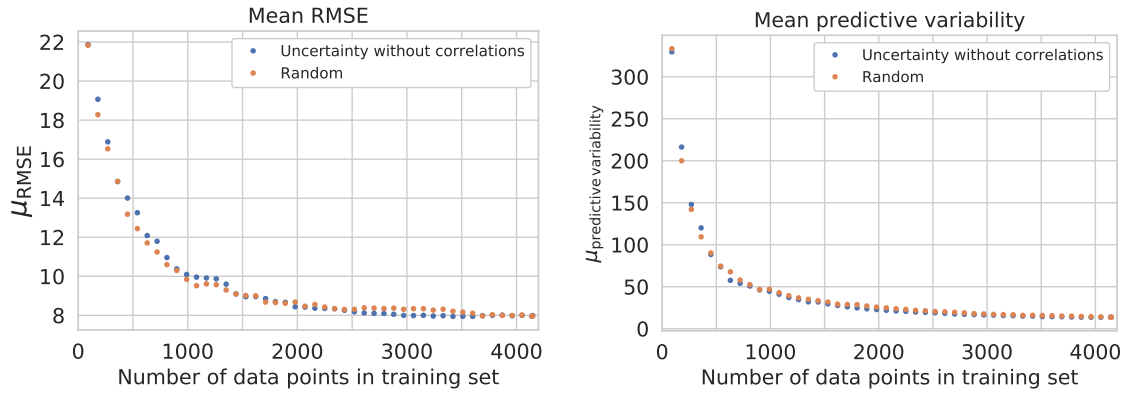
(a) Fold 1 **(b)** Fold 2

Figure A.13: The difference $\Delta\mu_{\text{predictive variability}} = \mu_{\text{predictive variability}}^{\text{uncertainty}} - \mu_{\text{predictive variability}}^{\text{random}}$ of the mean predictive variabilities of uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Pfizer data that have been added to the training set. Fold 1 and 2, of in total ten splits, of the cross validation are displayed. The red dotted line displays where the difference is equal to zero and the error bars display the approximate 95% confidence intervals.



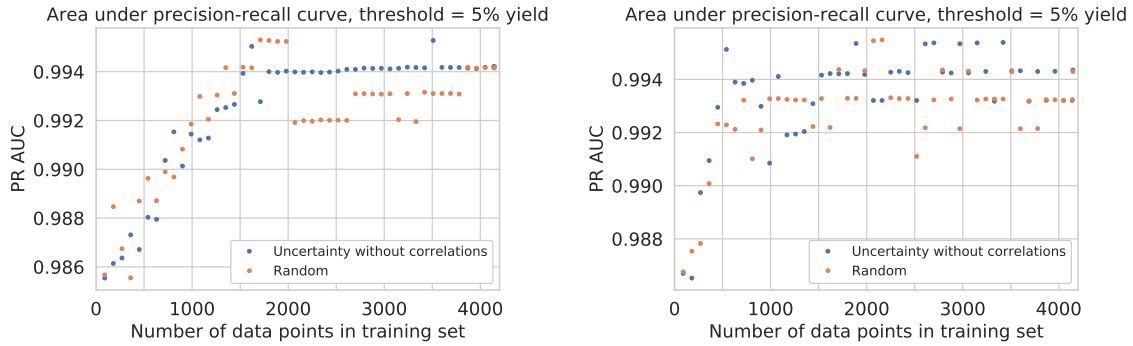
(a) RMSE **(b)** Predictive variability

Figure A.14: The average differences $\Delta\mu_{\text{RMSE}}$ and $\Delta\mu_{\text{predictive variability}}$ between uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Pfizer data that have been added to the training set. The red dotted line displays where the difference is equal to zero.



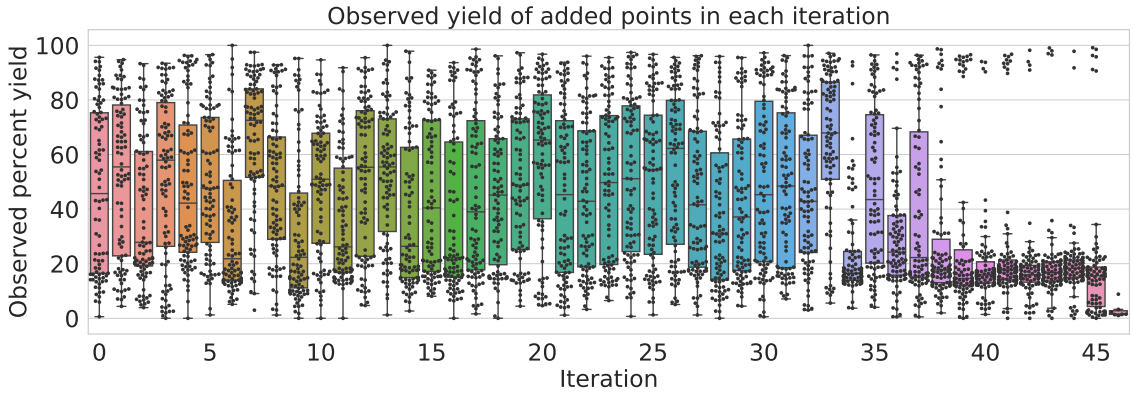
(a) RMSE **(b)** Predictive variability

Figure A.15: The mean RMSE and mean predictive variability of uncertainty sampling without correlation simulations and random sampling as a function of the number of points from the Pfizer data that have been added to the training set. Only the results for split 1 are shown, but the other splits show similar convergences.

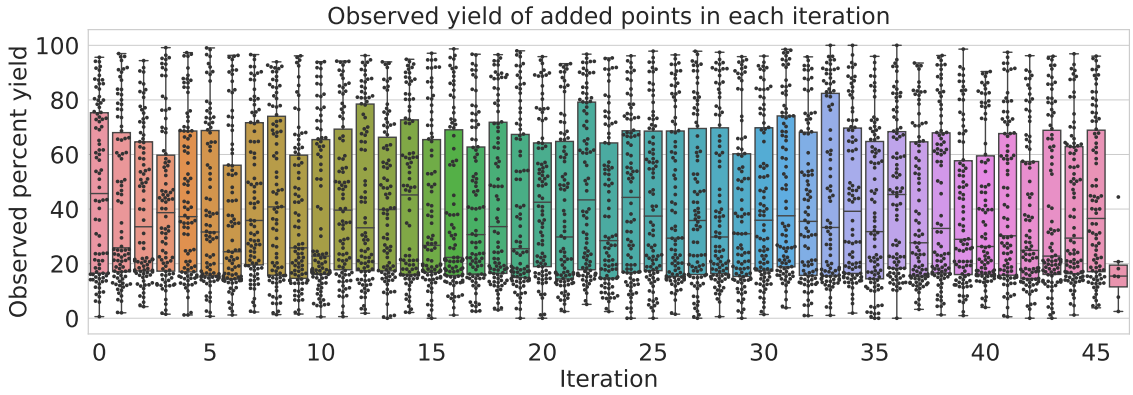


(a) Fold 1 **(b)** Fold 2

Figure A.16: Area under precision-recall curve of uncertainty sampling without correlation simulations and random sampling applied on the Pfizer data.

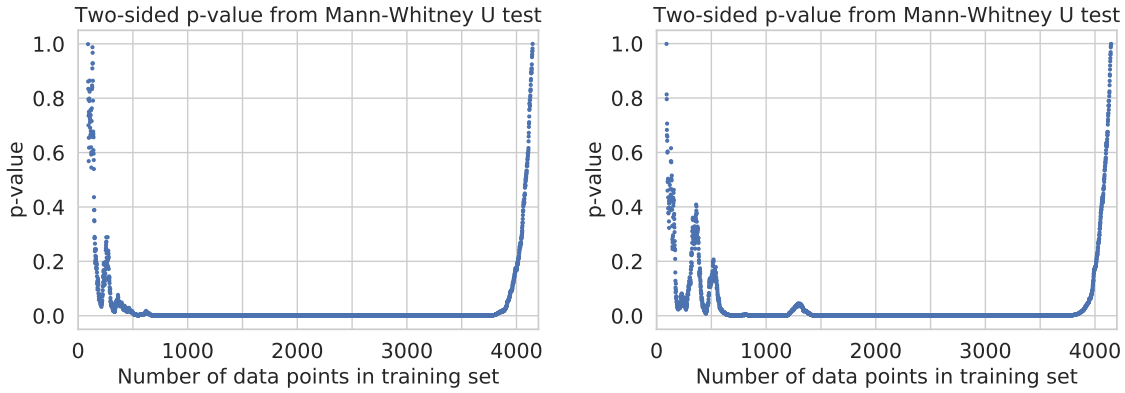


(a) Uncertainty sampling without correlation simulations



(b) Random sampling

Figure A.17: Observed percent yield of points from the Pfizer data that have been added to the training set in each iteration when utilizing uncertainty sampling without correlation simulations and random sampling. Only split 1 is displayed but the other splits show similar behaviors.



(a) Fold 1 **(b)** Fold 2

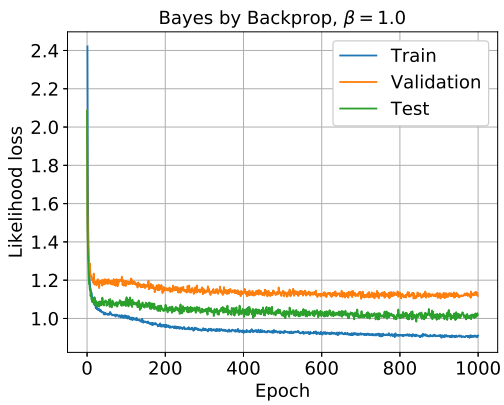
Figure A.18: p-values of a Mann-Whitney U test of all subsets of the training set with points from the Pfizer data when utilizing uncertainty sampling without correlation simulations and random sampling. p-values are not plotted for the subsets of the initialization. Only split 1 and 2 are displayed but the other splits show similar behaviors.

A.2 AZ ELN Performance

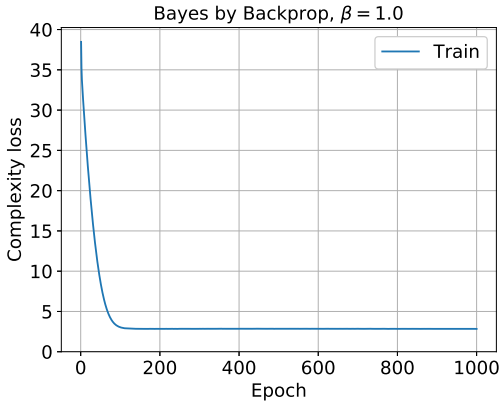
This section presents the predictive performance on the second CV fold for the Bayesian neural networks.

A.2.1 Bayes by Backprop

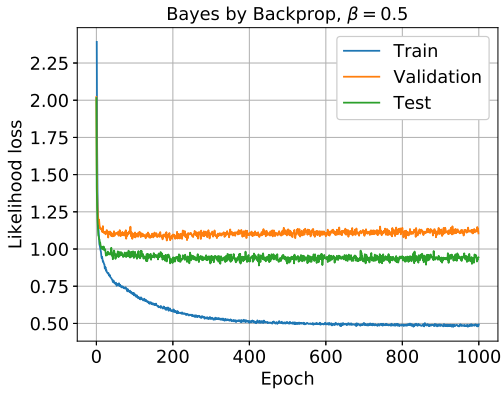
This section presents the predictive performance on the second CV fold for the Bayes by Backprop approach. The remaining folds illustrate similar trends. Note in particular, that these results correspond well to the results presented in Section 4.2.1.



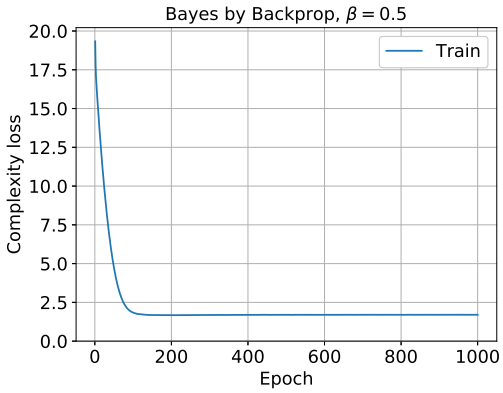
(a) Likelihood loss (Negative log likelihood)



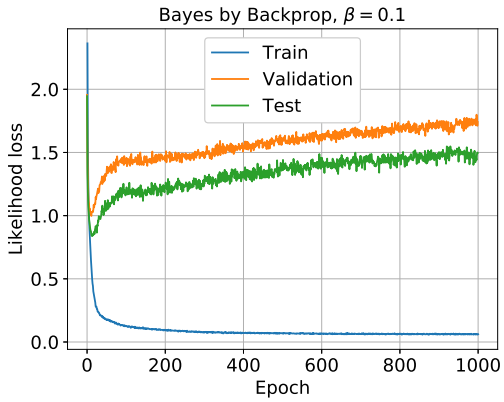
(b) Complexity loss



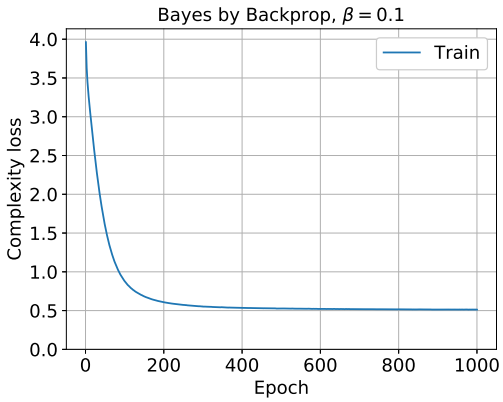
(c) Likelihood loss (Negative log likelihood)



(d) Complexity loss

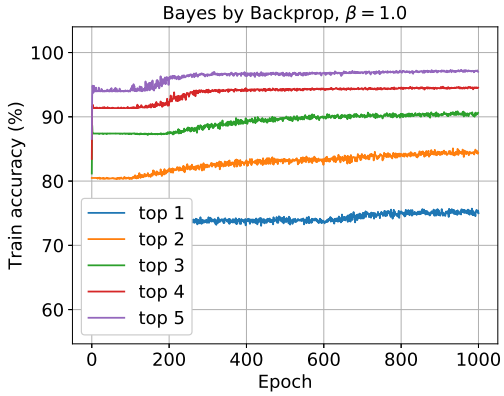


(e) Likelihood loss (Negative log likelihood)

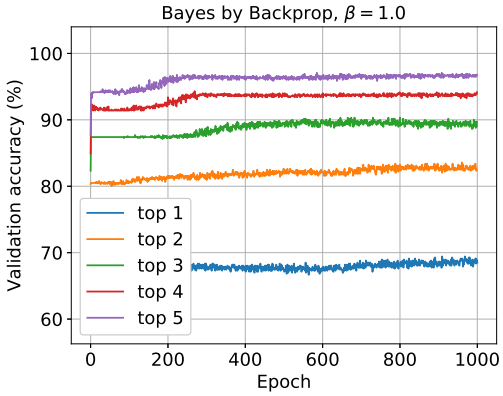


(f) Complexity loss

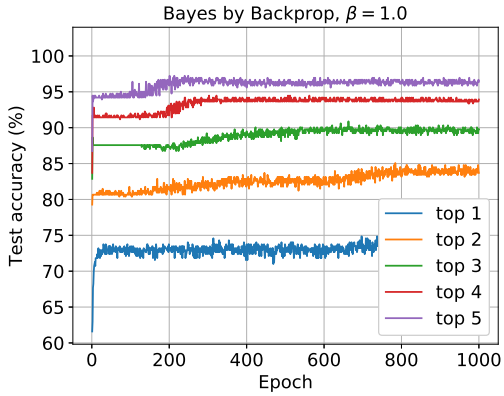
Figure A.19: The likelihood and complexity loss on the second CV fold of the AZ ELN data. The loss is averaged over the number of data points in each set. Note that since the complexity loss does not depend on the data it is constant between the train, validation and test split. Furthermore, note that the training likelihood loss decreases with β.



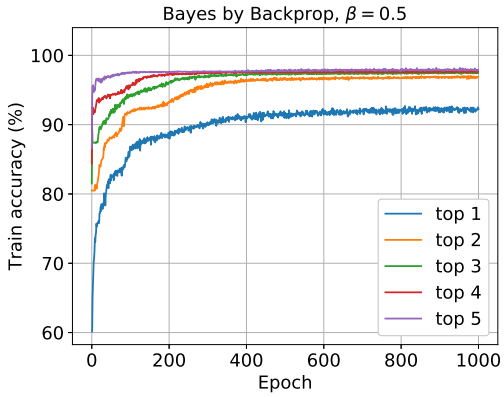
(a) Train



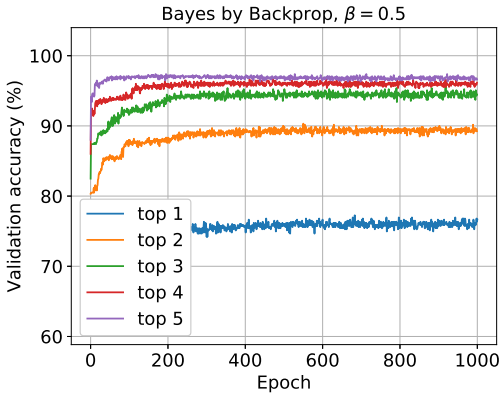
(b) Validation



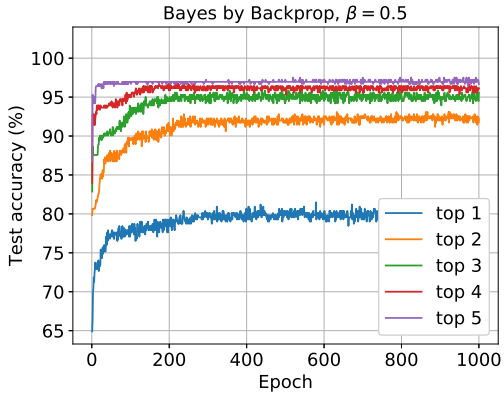
(c) Test



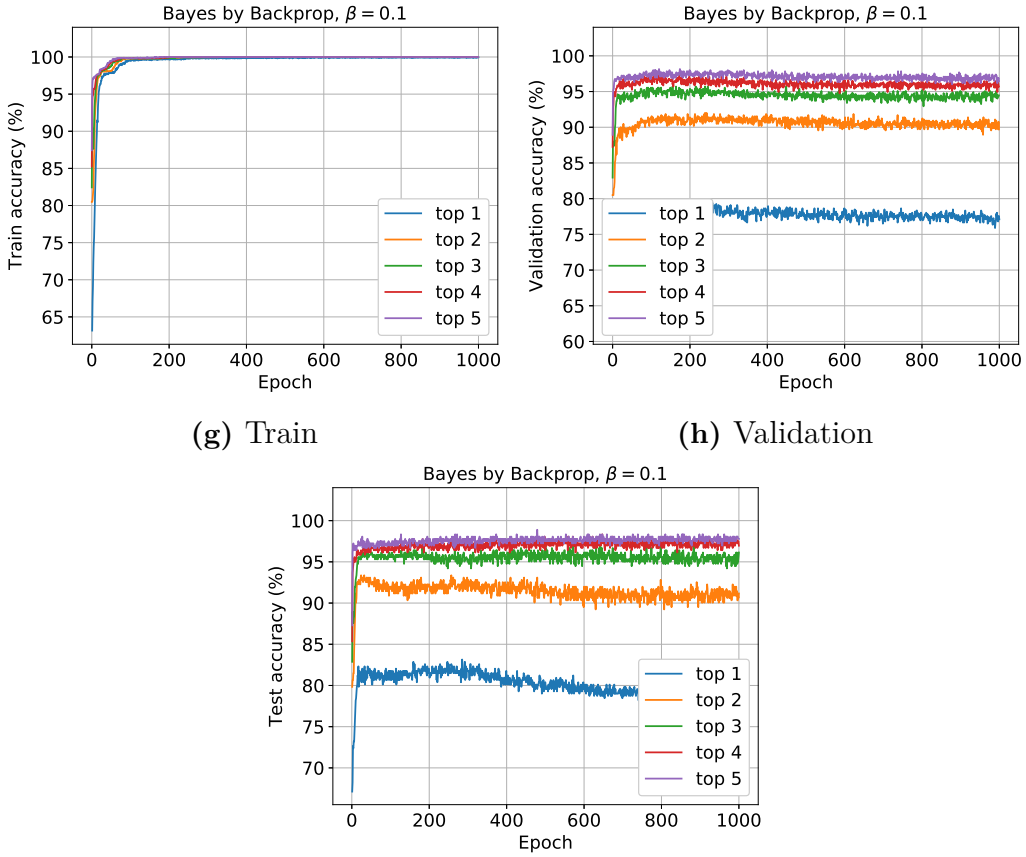
(d) Train



(e) Validation



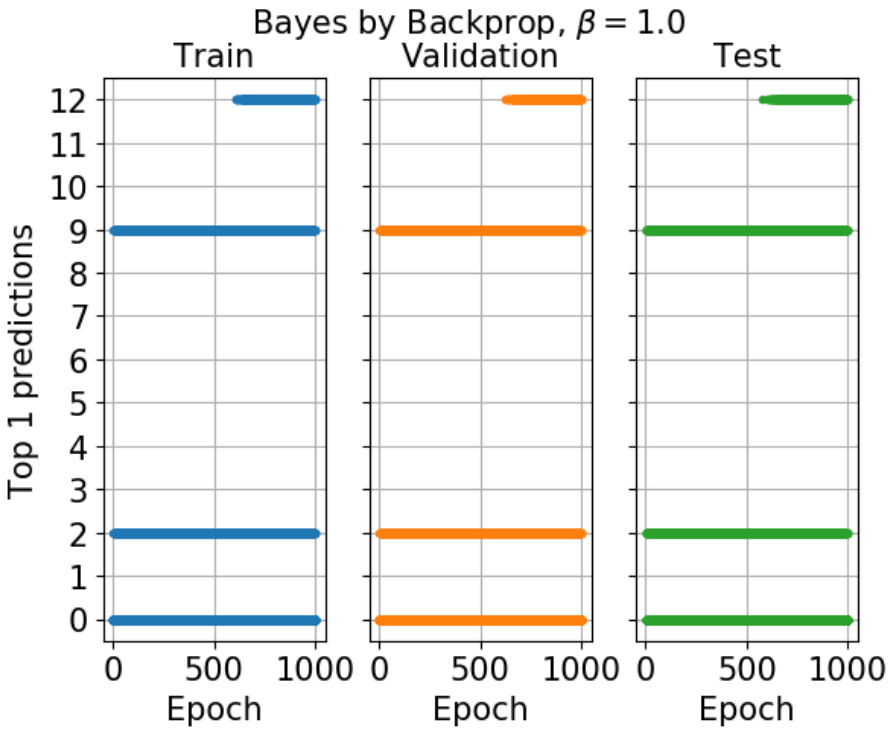
(f) Test



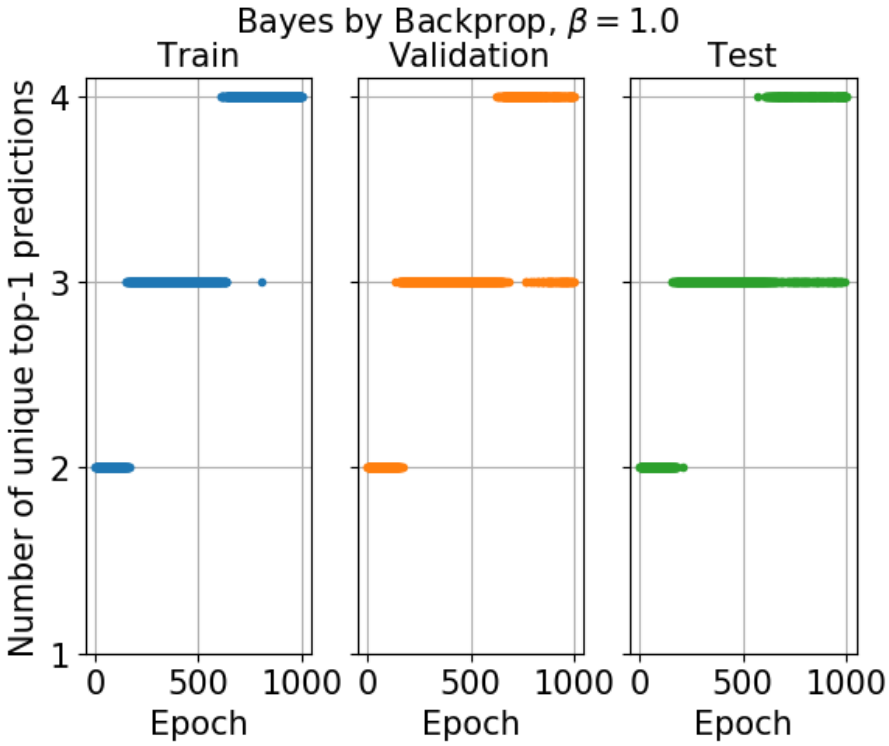
(g) Train **(h)** Validation

(i) Test

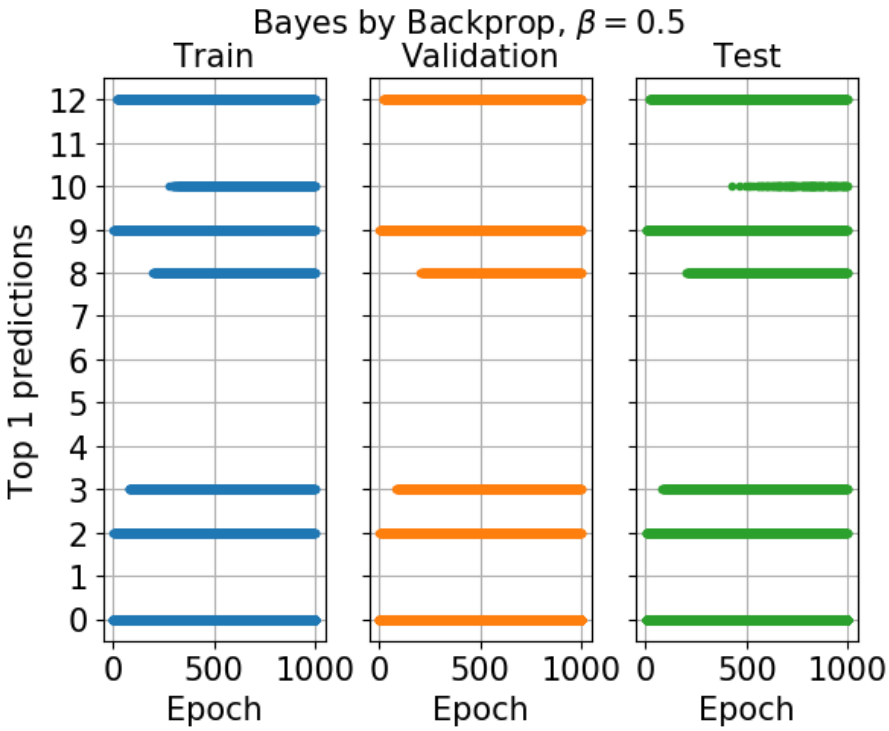
Figure A.20: The top n accuracy on the second CV fold of the AZ ELN data. Note that for $\beta = 1.0$ the performance is very similar on the training, validation and test set. However, for $\beta \in \{0.1, 0.5\}$ the training accuracy is significantly higher. Finally, note that the top 5 accuracy for the validation and test set is very similar for all values of β.



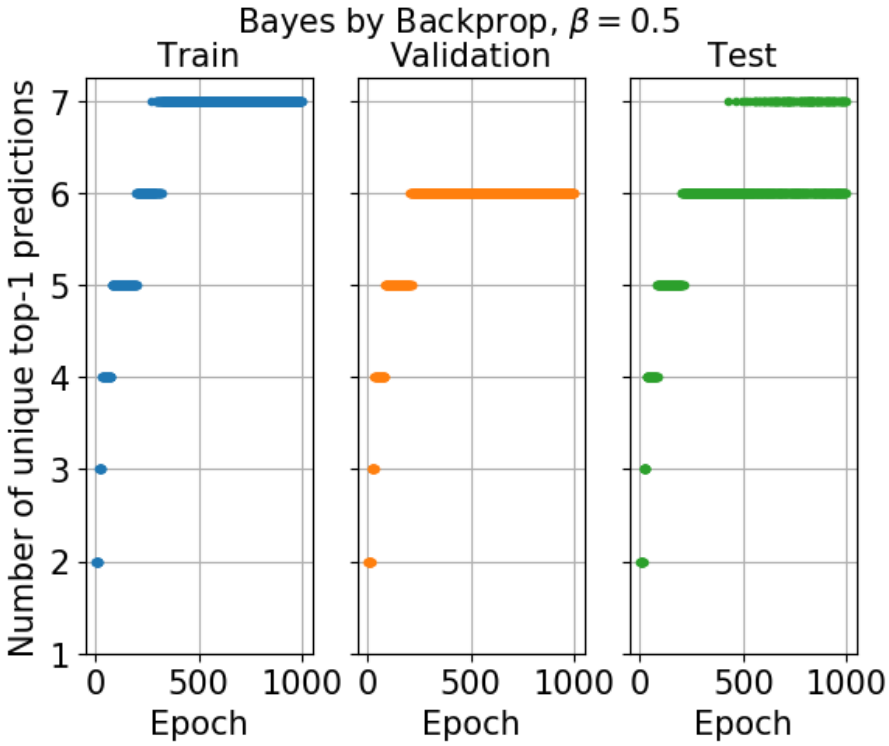
(a) Predicted ligands



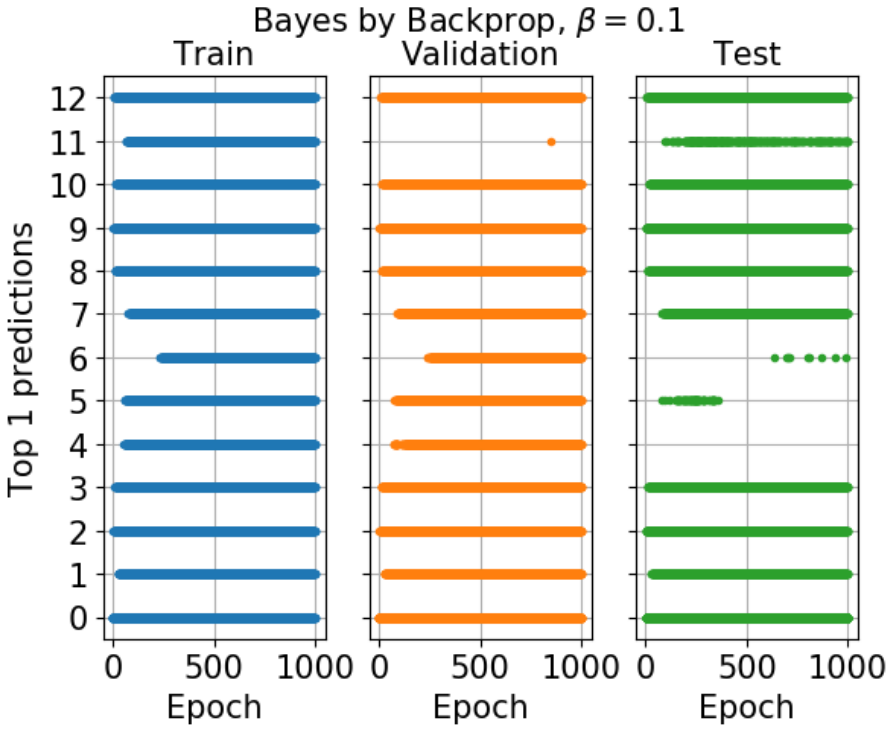
(b) Number of predicted ligands



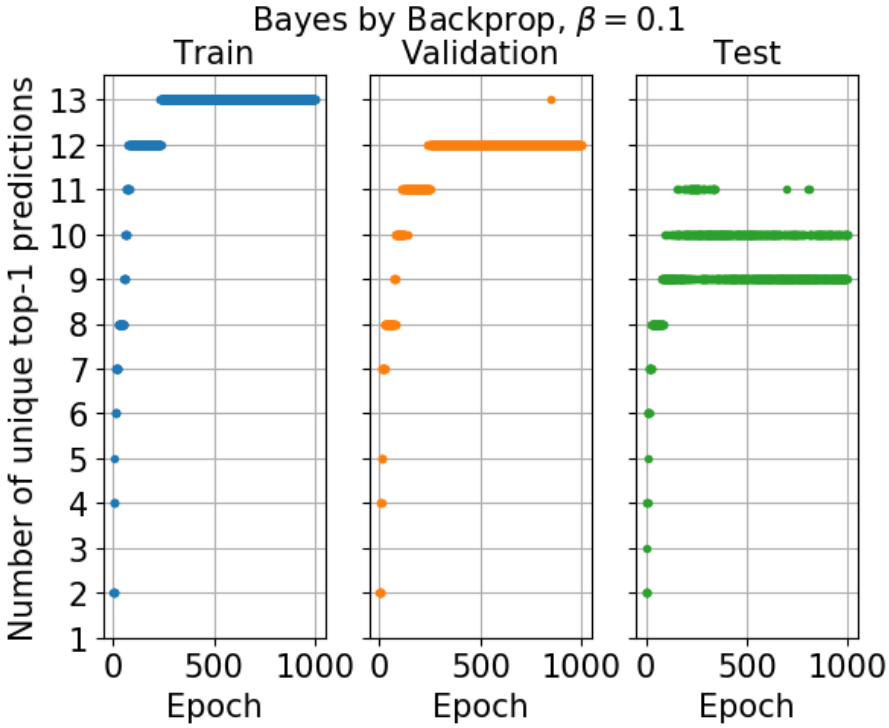
(c) Predicted ligands



(d) Number of predicted ligands



(e) Predicted ligands

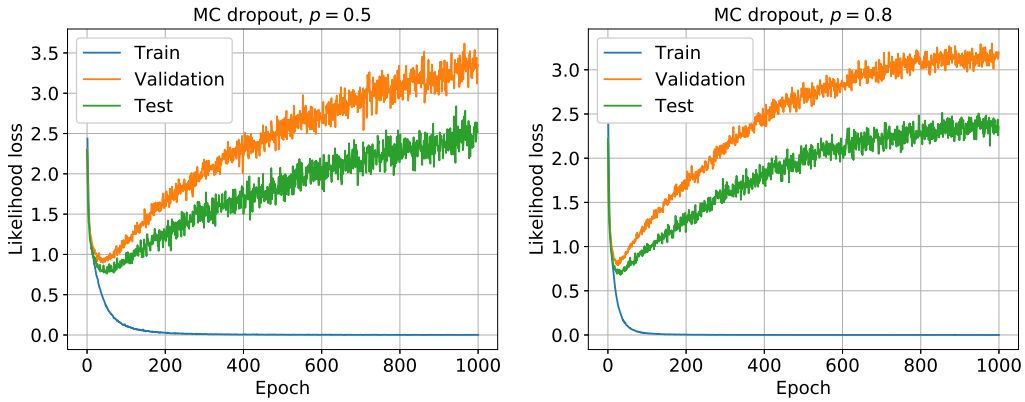


(f) Number of predicted ligands

Figure A.21: Predicted ligands and number of predicted ligands on the second CV fold of the AZ ELN data. Predicted here means that the ligand is the most frequently observed outcome of the network based on 100 forward passes of the same input. Note that the trend between the three splits is very similar for $\beta = 1.0$.

A.2.2 MC Dropout

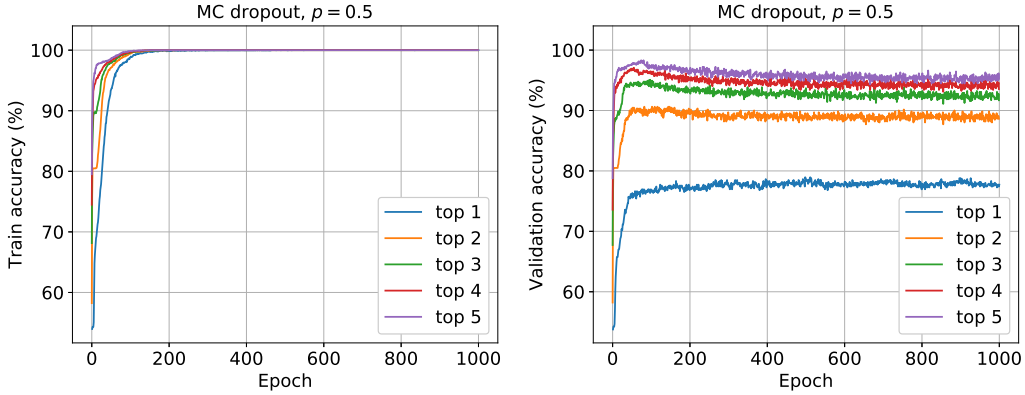
This section presents the predictive performance on the second CV fold for the MC dropout approach. The remaining folds (not presented in this thesis) illustrate similar trends to this fold. Note that the overfitting for this folds, and hence the remaining folds, is more severe than the overfitting on the first fold (Section 4.2.2). However, it is worth noting that $p = 0.8$ still overfits the data less than $p = 0.5$, and that the likelihood loss appears to stagnate after around 1000 epochs. Lastly, due to the more severe overfitting on these folds the model based on fold 1 was used to investigate the uncertainty.



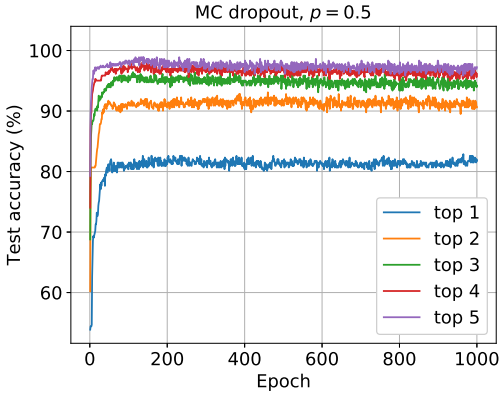
(a) Likelihood loss (Negative log likelihood)

(b) Likelihood loss (Negative log likelihood)

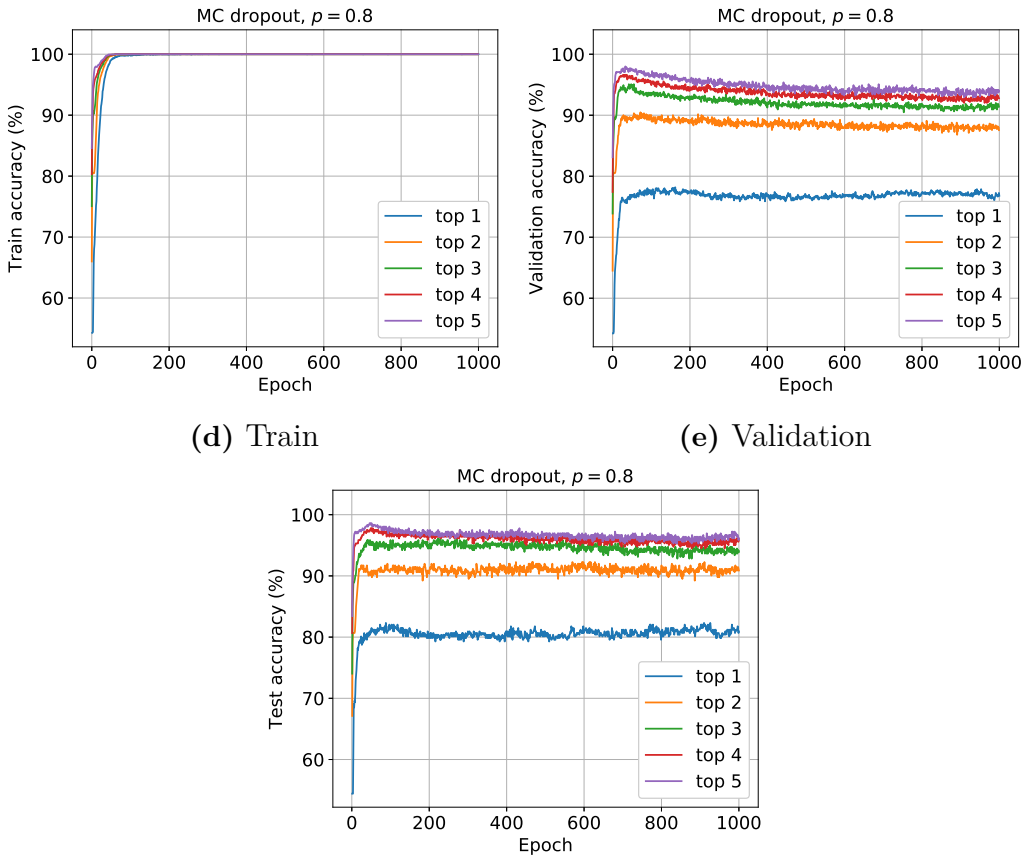
Figure A.22: The likelihood loss for MC dropout on the second CV fold of the AZ ELN data. The loss is averaged over the number of data points in each set. Note that after the first 50 epochs the model starts to overfit the data for both values of p.



(a) Train **(b)** Validation



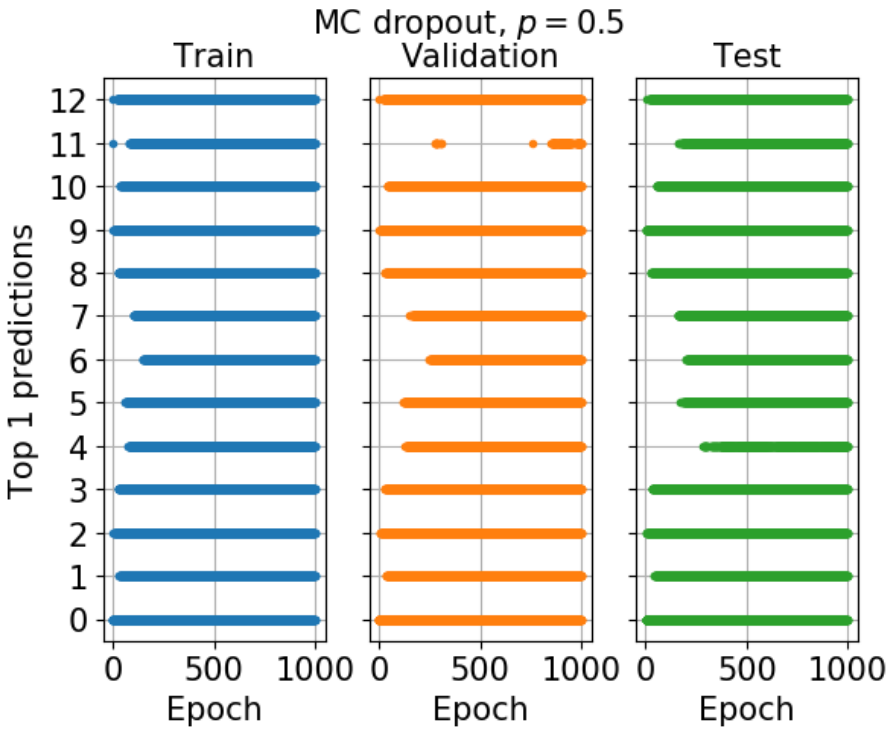
(c) Test



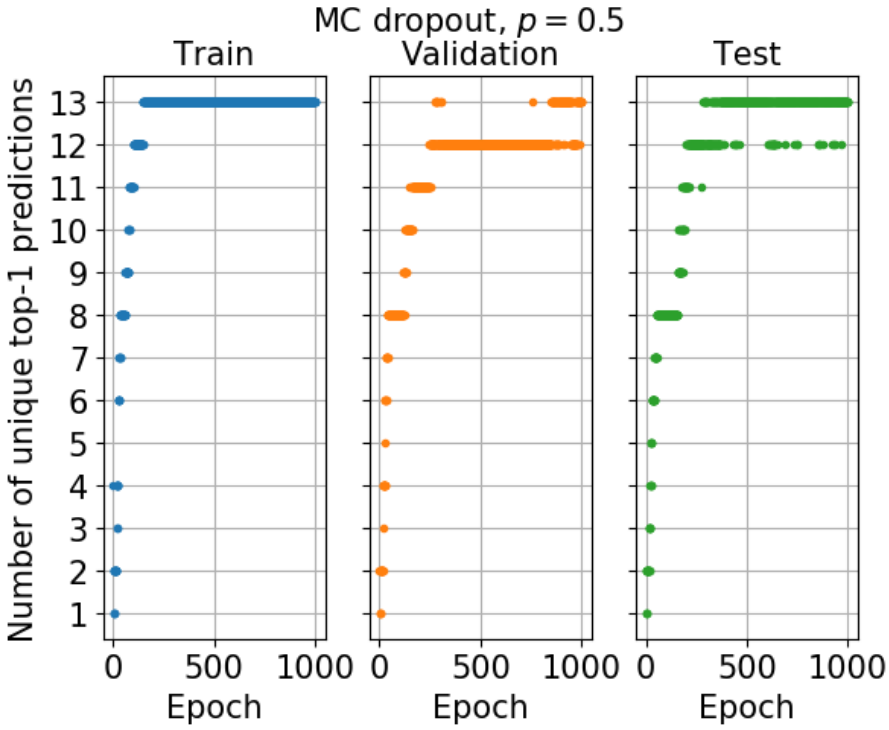
(d) Train **(e)** Validation

(f) Test

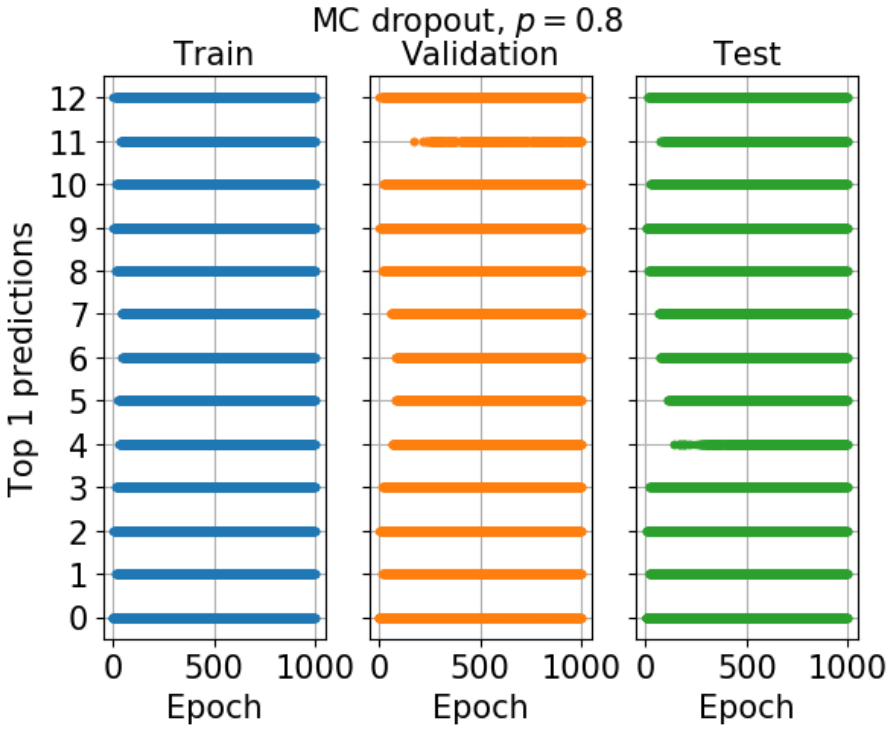
Figure A.23: The top n accuracy for MC dropout on the second CV fold of the AZ ELN data. Note that for both values of p the top n accuracy for the training set is significantly higher than the corresponding accuracy for validation and test split.



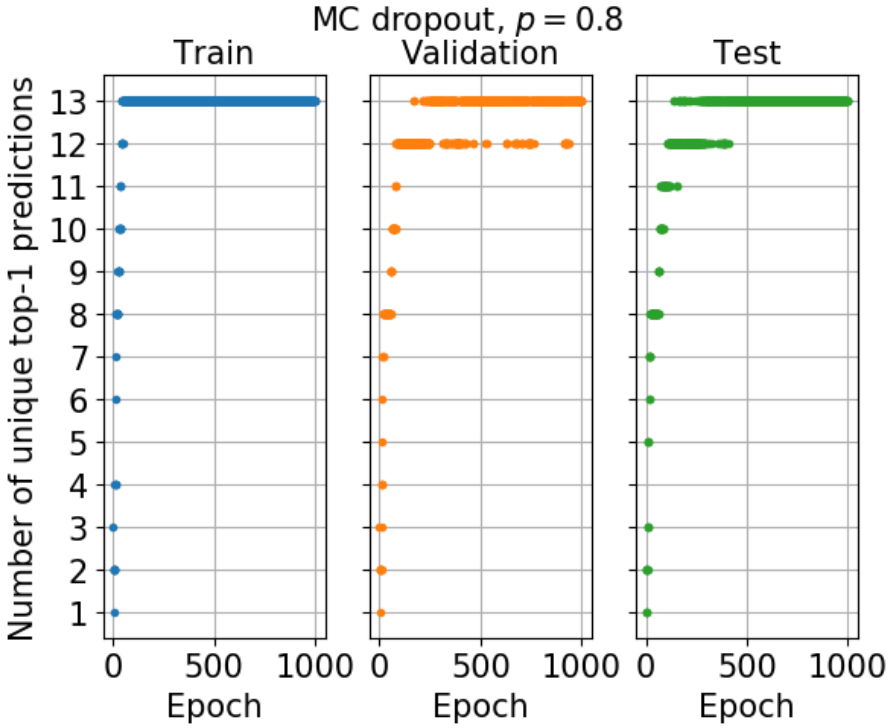
(a) Predicted ligands



(b) Number of predicted ligands



(c) Predicted ligands



(d) Number of predicted ligands

Figure A.24: Predicted ligands and number of predicted ligands for the second CV fold of the AZ ELN data. Predicted here means that the ligand is the most frequently observed outcome of the network based on 100 forward passes of the same input. Note that the trend between the validation and test split is very similar.