# Inaugural – Dissertation

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich-Mathematischen Gesamtfakultät

der

# Ruprecht-Karls-Universität Heidelberg

vorgelegt von

Manuel Haußmann, M.Sc.

geboren in Stuttgart, Deutschland

Tag der mündlichen Prüfung:

27.07.2021

# BAYESIAN NEURAL NETWORKS FOR PROBABILISTIC MACHINE LEARNING

MANUEL HAUSSMANN

Advisor: Prof. Dr. Fred A. Hamprecht

# ABSTRACT

Deep Learning-based models are becoming more and more relevant for an increasing number of applications. Bayesian neural networks can serve as a principled way to model the uncertainty in such approaches and to include prior knowledge. This work tackles how to improve the training of Bayesian neural nets (BNNs) and how to apply them in practice. We first develop a variational inference-based approach to learn them without requiring samples during training using the popular rectified linear unit activation function's piecewise linear structure. We then show how we can use a second approach based on a central limit theorem argument to get a good predictive uncertainty signal for an active learning task. We further build a reinforcement learning-based approach in such an active learning setup, learning a second BNN that requests labels to support the primary model optimally. As a third variant, we then introduce a new method for learning BNNs by optimizing the marginal likelihood via a model selection based approach, relying on the concept of type-II maximum likelihood, also known as empirical Bayes. Using PAC-Bayes theory to develop a regularization structure, we show how to combine it with a popular deterministic model for out-of-distribution detection, demonstrating improved results. Using this joint combination of empirical Bayes and PAC-Bayes, we finally study how to use it to learn dynamical systems specified via stochastic differential equations in a way that allows incorporating prior knowledge of the dynamics and model uncertainty.

# ZUSAMMENFASSUNG

Deep Learning-basierte Modelle werden für eine zunehmende Anzahl von Anwendungen immer relevanter. Bayes'sche neuronale Netze können als prinzipielle Möglichkeit dienen, die Unsicherheit in solchen Ansätzen zu modellieren und Vorwissen einzubeziehen. Diese Arbeit befasst sich damit, wie das Training von Bayes'schen Neuronalen Netzen (BNNs) verbessert werden kann und wie sie in der Praxis eingesetzt werden können. Wir entwickeln zunächst einen auf Variationsinferenz basierenden Ansatz, um sie zu lernen, ohne dass während des Trainings Stichproben benötigt werden, indem wir die stückweise lineare Struktur der beliebten ReLU Aktivierungsfunktion verwenden. Dann zeigen wir, wie wir einen zweiten Ansatz verwenden können, der auf einem Argument des zentralen Grenzwertsatzes basiert, um ein gutes Vorhersageunsicherheitssignal für das Problem des Aktiven Lernens zu erhalten. Weiterhin konstruieren wir einen auf Reinforcement Learning basierenden Ansatz für dieses Problem, welches ein zweites BNN lernt, um das primäre Modell optimal zu unterstützen. Als dritte Variante führen wir dann eine neue Methode zum Lernen von BNNs ein, indem wir die marginale Wahrscheinlichkeit über einen auf Modellauswahl basierenden Ansatz optimieren, der sich auf das Konzept eines Maximum-Likelihood-Schätzers vom Typ 2 stützt, auch bekannt als empirical Bayes. Unter Verwendung der PAC-Bayes-Theorie zur Entwicklung einer Regularisierungsstruktur zeigen wir, wie diese mit einem populären deterministischen Modell zur Erkennung von Out-of-Distribution kombiniert werden kann, und demonstrieren verbesserte Ergebnisse. Unter Verwendung dieser gemeinsamen Kombination von empirical Bayes und PAC-Bayes untersuchen wir schließlich wie man damit dynamische Systeme, die durch stochastische Differentialgleichungen spezifiziert sind, auf eine Weise lernen kann, die es erlaubt, Vorwissen über die Dynamik und Modellunsicherheit einzubeziehen.

# ACKNOWLEDGMENTS

My thanks belong first of all to Fred and the whole Image Analysis and Learning group. Without all of you giving me a home for the last couple of years and letting me explore my crazy Bayesian ideas, none of this thesis would exist. Thanks are especially due to Dominik and Sven for being great officemates in building B, and the many shared non-Mensa lunchbreaks and Tischkicker games. Thanks also to Elke for sharing the office with me in building A, tolerating my constant stream of PhD-comics, and ensuring that the group had regular cake breaks. Thanks to Ulli for many spontaneous brainstorming session whenever we met on the staircases of the Mathematikon. Special thanks also to Barbara for keeping the group running through the daily kafkaesque bureaucratic struggles.

Thanks are due to Dr. Sawitzki. Your small introductory course into the strange, interesting world of computational statistics started the journey of getting me interested in that direction. The same thanks go to the many professors and scientists who put their courses, papers, and ideas out there in the free world for everybody to learn from. And, given that this thesis is written using LaTeX on a Linux distribution, its experiments having been mostly coded in python, thanks is due to the community and the many volunteers who provide time and code and keep this whole infrastructure running.

Finally, a huge and major thank you is due to Melih. Your course on "Bayesian Modeling and Inference" took me fully into the world of Machine Learning and Bayes. When we started the discussion about a master thesis, you offered me the choice between two paths. The safe path that ensures a clean master thesis but without being too interesting, and the interesting research path, full of head scratching and potential dead ends if ideas don't work out, but if they do, leading to real knowledge and progress. I think with hindsight, we can say that we picked the right path. Thank you for guiding me through these last years. Let's see where the path leads to in the future!

Lastly, yes, there is indeed a thank you after the final one, due to my family being out of competition: Thanks for being the best family around and supporting me and keeping me sane whenever I think about starting some strange project, such as pursuing a PhD.

# CONTENTS

# 1

# INTRODUCTION

The theory of neural networks has witnessed several waves of popularity since its inception more than half a century ago. Under the keyword *"Deep Learning"* the current wave is larger than ever demonstrating its usefulness in a broad range of applications as diverse as computer vision (He et al., 2015), protein folding (Jumper et al., 2020), game playing (Silver et al., 2018), and text generation (Brown et al., 2020).

These advances offer not only great potential for society but also pose new questions. One is the similarly great potential of ethical danger and problems, which we as researchers should always keep at the back of our minds instead of simply stating "That's not my department", as Tom Lehrer so wonderfully summarized in his classic song[1]. These problems will, however, not be the topic of this thesis.

Instead, we target the question of how to use their probabilistic counterparts in the form of Bayesian Neural Networks (BNNs). In their ideal form, they—and Bayesian models in general (Ghahramani, 2015)—offer a principled way to improve the feedback the researcher gets from neural nets from simple point predictions to well-calibrated predictive uncertainties, indicating when one can trust such a model. They allow for principled modelling of the uncertainty, whether lack of predictive certainty is due to inherent randomness or lack of observations. Additionally, they offer a principled way of performing model selection, i.e. the comparison between different models and how to incorporate prior knowledge instead of having to rely on purely black-box models.

BNNs have seen similar fluctuations in popularity as deterministic nets, from being extensively studied in the early nineties (MacKay, 1992, 1995; Neal, 1995), to being superseded in popularity by Gaussian processes (Rasmussen and Williams, 2006), which lead David MacKay to his popular question of whether we have thrown *"the baby out with the bathwater"* in that choice, or whether BNNs are justified in having their own parallel existence and benefits. Following the increase in Deep Learning research, BNNs have grown similarly in popularity again. However, the list of benefits compared to the deterministic approaches mentioned above raises the question of why they haven't simply replaced their deterministic counterparts? The answer is hidden in the constraint "in their ideal form". While the ever-increasing computational power helps a lot, the problem of there being "no free lunch" still applies. In practice, Bayesian

1 The quote is from Tom Lehrer's song *Wernher von Brown*, see e.g. `https://tomlehrersongs.com/wernher-von-braun/` for the full lyrics.

neural networks are still a lot more challenging to train, requiring simplifying approximations to be usable.

This thesis aims to contribute to the growing body of literature that seeks to answer MacKay's question affirmatively, developing new methods of training them more efficiently and exploring how to use them in various applications. It is organized as follows.

The next chapter starts the journey by giving a high-level overview of the areas and concepts required for the rest of the thesis, pointing to further reading material for each of them. Its second task is to introduce the notation used throughout. A summary of the notation, as well as details on the distributions, special functions and inequalities, are also provided in the appendix (Chapter 8). This chapter can also be skipped and referred to whenever necessary.

*I am a marginal note*

Throughout the thesis, we use marginal notes to allow for easy visual guidance, which helps the reader to jump to relevant parts quickly.

In Chapter 3 we introduce a new family of variational posterior distributions for BNNs. Using the popular rectified linear unit (ReLU) activation, we can decompose the forward pass, allowing us to learn a BNN without requiring samples of the weights, as most common variational inference-based approaches do. Additionally, this structure will enable us to learn part of the variational variables via closed-form updates, further improving the learning procedure and demonstrate its usefulness and competitiveness on several standard regression and classification tasks.

A second approach for (mostly) sampling-free learning is explored in Chapter 4. We demonstrate that we can use a central limit theorem based variational inference approach to get good predictive uncertainties. We use and evaluate them in the application area of Active Learning. The second contribution in that chapter is that we develop a way to replace the commonly hard-coded and fixed acquisition function necessary for the active learning task, with a second BNN that learns how to request new labels during the active learning process, adapting itself to the specifics of the data set at hand.

We pursue this central limit theorem based approach further in Chapter 5 where we introduce a new way to learn Bayesian neural nets, by formulating the training objective as an empirical Bayes/type-II Maximum Likelihood. We show that this allows us to combine the model with evidential deep learning (Sensoy et al., 2018) extending it to be able to distinguish between reducible (epistemic) uncertainty and irreducible (aleatoric) uncertainty. As this introduces a wide range of hyperparameters, we show how to adapt PAC-Bayes generalization bounds to our setup to get a reliably regularizing signal, showing improved performance of the final model in the task of out-of-distribution detection.

In Chapter 6 we consider another application that is learning stochastic dynamics assuming some potential prior knowledge. We develop a way to learn such stochastic differential equation-based models via Bayesian neural networks by

deriving another PAC-Bayes based objective that allows for the incorporation of prior knowledge.

### PUBLICATIONS AND FURTHER COLLABORATIONS

This thesis is based on the following four publications/preprints:

- Manuel Haußmann, Fred A. Hamprecht, and Melih Kandemir (2019). "Sampling-Free Variational Inference of Bayesian Neural Networks by Variance Backpropagation". *Proceedings of the Thirty-Fifth Uncertainty in Artificial Intelligence Conference*. (Chapter 3)
- Manuel Haußmann, Fred A. Hamprecht, and Melih Kandemir (2019). "Deep Active Learning with Adaptive Acquisition". *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. (Chapter 4)
- Manuel Haußmann, Sebastian Gerwinn, and Melih Kandemir (2020). "Bayesian Evidential Deep Learning with PAC Regularization". *3rd Symposium on Advances in Approximate Bayesian Inference*. (Chapter 5)
- Manuel Haußmann*, Sebastian Gerwinn*, Andreas Look, Barbara Rakitsch, and Melih Kandemir (2021) "Learning Partially Known Stochastic Dynamics with Empirical PAC Bayes". *Proceedings of the Twenty-Fourth International Conference on Artificial Intelligence and Statistics*. (Chapter 6)

The author further contributed to the following publications:

- Elke Kirschbaum, Manuel Haußmann, Steffen Wolf, Hannah Sonntag, Justus Schneider, Shehabeldin Elzoheiry, Oliver Kann, Daniel Durstewitz, and Fred A. Hamprecht (2019). "LeMoNADe: Learned Motif and Neuronal Assembly Detection in Calcium Imaging Videos". *International Conference on Learning Representations*
- Sven Bollweg, Manuel Haußmann, Gregor Kasieczka, Michel Luchmann, Tilman Plehn, and Jennifer Thompson (2020). "Deep-learning Jets with Uncertainties and More ". *SciPost Phys. 8, 006*

* Equal contribution

# 2

# BACKGROUND

Throughout this chapter, we will introduce and give a high-level overview of the background material required for the results discussed in the rest of the thesis. This chapter serves a second goal: the discussion of background material will also allow us to use this chapter as an introduction and reference for the notation used throughout the thesis.

The following sections may be skimmed, or skipped upon first reading and referred to whenever necessary. They aim to give intuition to the most relevant topics needed to properly understand the following chapters, with references to the literature for further background reading. The highlights in the margins serve as guidance to quickly locate the relevant parts.

In the first section, we first discuss the Bayesian approach to learning, focusing on our primary tool, variational inference. Section 2.2 gives a quick introduction to neural networks, together with introducing the notation we will rely on throughout the work. Afterwards, we discuss the concept of active learning in Section 2.3. Section 2.4 gives an overview of reinforcement learning in the form of Markov decision processes and one way to learn them. In Section 2.5 we introduce generalization bounds as used in the PAC-Bayes approach, and finally close the chapter in Section 2.6 with stochastic differential equations and a numerical method to solving them. See also the appendix for a list of distributions and (in)equalities used throughout this work.

## 2.1 BAYESIAN MACHINE LEARNING AND VARIATIONAL INFERENCE

The thesis sails under the flag of Bayesian probabilistic machine learning. This raises the question, what do we mean by that? And how does our understanding fit in with the greater literature? We will not be able to give a thorough treatment of the whole of the current Bayesian literature and how it is used in machine learning, nor is this the right place for such an undertaking. Instead, we focus on the relevant terms required for the later chapters, focusing primarily on the area of variational inference.

For a general introduction to the Bayesian approach, the book by McElreath (2020) offers an excellent practical introduction focusing on the area of Bayesian statistics. For a useful follow-up Gelman et al. (2013) give a thorough theoretical treatment. For probabilistic and Bayesian approaches to Machine Learning Ghahramani (2015) gives a general overview, and there exist a couple of great textbooks for a detailed introduction (MacKay, 2003; Bishop, 2006; Barber, 2012; Murphy, 2012).

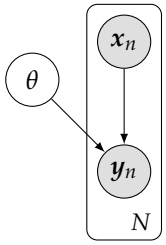



**Figure 2.1: A general generative model.** We use filled circles for observed variables $\boldsymbol{y}_n, \boldsymbol{x}_n$ and unfilled circles for latent ones $(\theta)$. The plate notation indicates that there are $N$ pairs $(\boldsymbol{x}_n, \boldsymbol{y}_n)$.

Relevant for the following chapters is an understanding of the theory on the following level. We always assume to be in the case of supervised learning throughout this work. The variables in the models can then be grouped into three broad groups. We assume to have observed features $\boldsymbol{x}$, their targets $\boldsymbol{y}$,[1] and latent parameters $\theta$ governing their relationship. The generative model is given in its most general form as summarized in Figure 2.1

*Generative Model*

$$\begin{aligned} \theta &\sim p(\theta) \\ \boldsymbol{y}_n|\theta, \boldsymbol{x}_n &\sim p(\boldsymbol{y}_n|\theta, \boldsymbol{x}_n) \qquad i = 1, \ldots, N, \end{aligned}$$

i. e. we have a prior $p(\theta)$, and a conditional $p(\boldsymbol{y}|\boldsymbol{x}, \theta)$. Note that we won't model a distribution over the features $\boldsymbol{x}$, but still include them in terms of notation in the conditional probability for $\boldsymbol{y}$ to show their relationship. One could also extend most of the proposed approaches in the following chapters to accommodate for probabilistic modelling of these $\boldsymbol{x}$, which we will highlight whenever relevant.

Having observed $N$ pairs $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)$ we will refer to them collectively as $\mathcal{D}$ and use the shorthand $p(\mathcal{D}|\theta) = \prod_{n=1}^{N} p(\boldsymbol{y}_n|\theta, \boldsymbol{x}_n)$. The goal of inference/learning in a Bayesian setting is then to find the posterior distribution over the latent variables after having observed the data $\mathcal{D}$, i.e. $p(\theta|\mathcal{D})$, which is given via Bayes rule as [2]

*Inference via Bayes rule*

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \tag{2.1}$$

At test time, i.e. during the application of the learned model, the optimal posterior predictive distribution for $\boldsymbol{y}_*$ given the features $\boldsymbol{x}_*$ is then given as

*Prediction*

$$p(\boldsymbol{y}_*|\boldsymbol{x}_*, \mathcal{D}) = \int p(\boldsymbol{y}_*|\theta, \boldsymbol{x}_*)p(\theta|\mathcal{D})\,\mathrm{d}\theta. \tag{2.2}$$

So far for the theory. Unfortunately, for both inference and prediction, we are not able for most practically relevant models to come up with closed-form

1 Depending on the researcher's context and socialization, the independent $\boldsymbol{x}$ and the dependent $\boldsymbol{y}$, are known under a wide variety of expressions. We will mostly stick to feature/target throughout the thesis sometimes lapsing into input/output as is popular in the deep learning literature.

2 Note that in the deep learning literature the expression "inference" is often also applied to the task of prediction, leaving the task mentioned above as either learning or without a term at all.

expressions for each of the two distributions. For inference the marginal $p(\mathcal{D})$ is usually intractable, such that we cannot get an analytical posterior. Even if we were to, the integral in the predictive step usually remains intractable, requiring further approximations.

To solve the problem of the intractability of the posterior, there are two broad classes of approaches. First, for inference we can rely either on a sampling-based approach (Robert and Casella, 2013), or, what we will do instead use to ensure scalability to deep networks is to rely on Variational Inference (Wainwright and Jordan, 2008; Blei et al., 2017). We will introduce it in the next section, but first, close this section by focusing specifically on the term $p(\mathcal{D})$.

*Evidence and model selection*

EVIDENCE AND MODEL SELECTION This marginal $p(\mathcal{D})$ is also known as the *evidence*. The notation used above hides that all these probabilities depend on the choice of model, or theory underlying the specified generative model. For as specific theory or hypothesis $\mathcal{H}$ it would be more precise to write

$$p(\theta|\mathcal{D},\mathcal{H}) = \frac{p(\mathcal{D}|\theta,\mathcal{H})p(\theta|\mathcal{H})}{p(\mathcal{D}|\mathcal{H})}. \tag{2.3}$$

The evidence $p(\mathcal{D}|\mathcal{H})$ is then the marginal likelihood the theory assigns to the observed data after having marginalized over the latent parameters

$$p(\mathcal{D}|\mathcal{H}) = \int p(\mathcal{D}|\theta,\mathcal{H})p(\theta|\mathcal{H})\,\mathrm{d}\theta.$$

Assuming we have access to this evidence, it opens a way to compare different approaches in a principled way to perform model selection. For example, to decide between two models $\mathcal{H}_1, \mathcal{H}_2$, we can, relying on Bayes theorem again, compare their ratio as

$$\frac{p(\mathcal{H}_1|\mathcal{D})}{p(\mathcal{H}_2|\mathcal{D})} = \frac{p(\mathcal{H}_1)}{p(\mathcal{H}_2)} \cdot \frac{p(\mathcal{D}|\mathcal{H}_1)}{p(\mathcal{D}|\mathcal{H}_2)}. \tag{2.4}$$

Here, the first ratio on the right-hand side gives the relative prior probability of each hypothesis. In contrast, the second provides a ratio of evidence terms given each of the two models. This comparison then gives a principled Occam's razor approach. A complex model can explain a lot of observed data but necessarily assigns a lower probability to each one compared to a more constrained model (Rasmussen and Ghahramani, 2001; MacKay, 2003). These relative differences, then allow us to compare the posterior likelihood of each of the two given that we have observed a specific $\mathcal{D}$.

This model comparison will in its generality not be directly relevant for us for model selection, rather only in offering a principled way of comparing the final performance of different approaches in the literature. However, as we will discuss in Chapter 5, it will become relevant in a more constrained form. That is when we do not want to compare two general models $\mathcal{H}_1$ and $\mathcal{H}_2$. Instead, consider a family $\mathcal{H}_\alpha$ of models controlled by hyperparameters $\alpha$. For example in the general generative model, we described above, assume that the prior depends on hyperparameters, for example consider it to be a normal

distribution with a mean fixed to zero, but a precision hyperparameter given by $\alpha$, that is

$$\begin{aligned} \theta &\sim \mathcal{N}(\theta|0, \alpha^{-1}), \\ \boldsymbol{y}_n|\theta, \boldsymbol{x}_n &\sim p(\boldsymbol{y}_n|\theta, \boldsymbol{x}_n), \qquad i = 1, \ldots, N. \end{aligned}$$

We can do posterior inference in that setup as before. However, instead of relying on some grid search over possible $\alpha$ values to determine an optimal one, $p(\mathcal{D}|\mathcal{H}_\alpha)$ offers a principled measure that we can optimize with respect to $\alpha$.

*Empirical Bayes*
*Type-II ML*

This approach is known as empirical Bayes or type-II Maximum Likelihood in the literature (Bishop, 2006; Murphy, 2012). It is popular for example in the area of Gaussian Processes for fitting the hyperparameters of kernels, for example the length-scale of an exponentiated quadratic kernel (Rasmussen and Williams, 2006).

### 2.1.1 *Variational Inference*

Abstractly speaking, *Variational Inference (VI)* tries to approximate the intractable $p(\theta|\mathcal{D})$ with a tractable distribution $q(\theta)$ from some family of distributions $Q$. Closeness between these two distributions is measured here in terms of the *Kullback-Leibler (KL) divergence* (also known as relative entropy in information theory),

*Kullback-Leibler divergence*

$$\begin{aligned} \mathrm{KL}\left(q(\theta) \parallel p(\theta|\mathcal{D})\right) &\triangleq -\int q(\theta) \log \frac{p(\theta|\mathcal{D})}{q(\theta)} \,\mathrm{d}\theta \\ &= -\mathbb{E}_{q(\theta)}\left[\log \frac{p(\theta|\mathcal{D})}{q(\theta)}\right] \\ &\geq -\log \int p(\theta|\mathcal{D}) \,\mathrm{d}\theta = 0, \end{aligned}$$

where the inequality uses Jensen's inequality. We use $\triangleq$ to refer to equality by definition. While it is not an actual distance (e.g. it lacks symmetry), it is constrained to be positive, and one can further easily show that it equals zero if and only if its arguments are equal. There are variations to this formulation, for example expectation propagation (Minka, 2001, 2013) which considers the opposite direction of the divergence, $\mathrm{KL}\left(p(\theta|\mathcal{D}) \parallel q(\theta)\right)$, or Rényi's $\alpha$-divergence based variational inference (Li and Turner, 2016), defined as

$$D_{\text{Rényi}}(p(\theta)||q(\theta)) = \frac{1}{\alpha - 1} \log \int p(\theta)^\alpha q(\theta)^{1-\alpha} \,\mathrm{d}\theta,$$

which in the limit of $\alpha \to 1$ recovers the KL divergence. However, we will stay with the version most commonly known and understood as variational inference.

*Evidence Lower Bound*

This objective in itself still requires the intractable posterior. However, we can make use of the following relationship by extending the log evidence as

$$\log p(\mathcal{D}) = \int q(\theta) \log p(\mathcal{D}) \,\mathrm{d}\theta$$

$$
\begin{aligned}
&= \int q(\theta) \log \frac{p(\mathcal{D}) q(\theta)}{q(\theta)} \, \mathrm{d}\theta \\
&= \int q(\theta) \log \frac{p(\theta, \mathcal{D}) q(\theta)}{p(\theta|\mathcal{D}) q(\theta)} \, \mathrm{d}\theta \\
&= \int q(\theta) \log \frac{p(\theta, \mathcal{D})}{q(\theta)} \, \mathrm{d}\theta + \mathrm{KL}\left(q(\theta) \parallel p(\theta|\mathcal{D})\right) \\
&\geq \int q(\theta) \log \frac{p(\theta, \mathcal{D})}{q(\theta)} \, \mathrm{d}\theta.
\end{aligned}
$$

The log marginal probability is independent of the $\theta$, giving us an upper bound. Simultaneously, the KL expression's positivity ensures that maximization of the resulting lower bound minimizes the desired objective. We will refer to this final expression as the *Evidence Lower BOund (ELBO)* throughout this thesis.

This leaves the choice of distributions as an open question. We will revisit the following two main approaches in the later chapters. The first is to specify a factorization of the $q(\theta)$ into $M$ groups of variables, such that $q(\theta) = \prod_{m=1}^{M} q_m(\theta_m)$. This factorization allows us to rewrite the ELBO, focusing on the $m$-th group, as

$$
\begin{aligned}
\int q(\theta) \log \frac{p(\theta, \mathcal{D})}{q(\theta)} \, \mathrm{d}\theta \overset{c}{=} &\int q_m(\theta_m) \mathbb{E}_{q_{\neq m}(\theta_{\neq m})} \left[\log p(\theta, \mathcal{D})\right] \mathrm{d}\theta_m \\
&- \int q_m(\theta_m) \log q_m(\theta_m) \, \mathrm{d}\theta_m,
\end{aligned}
$$

slightly abusing the notation with $q_{\neq m}(\theta_{\neq m}) = \prod_{i \neq m} q_i(\theta_i)$. We use $\overset{c}{=}$ to refer to equality up to a constant term. If we treat the other latent variables as fixed, this is a negative Kullback-Leiblere divergence. As it is maximized when both of its inputs are equal, the optimal solution $q_m^*$ is then given as

*Optimal updates via coordinate ascent*

$$
\log q_m^*(\theta_m) \overset{c}{=} \mathbb{E}_{q_{\neq m}(\theta_{\neq m})} \left[\log p(\theta, \mathcal{D})\right].
$$

If this expectation is analytically tractable, we have found an optimal update for each of the $M$ factors given the others. We can learn the variational posterior by a coordinate ascent approach, iteratively updating each factor until convergence. If that is possible, we have gained an efficient way of learning the variational posterior. If not, we have just exchanged one intractable objective for another. In Chapter 3 we will discuss one practical use case where these updates are indeed partially applicable. However, for most practical use cases with neural networks, this will not be the case requiring us to go one step further. In the following chapters, we will always rely on a full factorization, i.e. every variable is its own group, referred to as *mean-field*.

*Mean-field*

The second approach extends the first in specifying a factorization and an explicit parameterization of these distributions, for example by assuming the variational posterior to be an isotropic multivariate normal distribution with mean and variance parameters to be determined. Assuming that this variational

posterior is parameterized by $\phi$, the objective becomes a maximization of the ELBO, giving the optimal choice of parameters as

$$\phi^* = \arg\max_{\phi} \mathbb{E}_{q_\phi(\theta)} \left[\log p(\mathcal{D}|\theta)\right] + \mathrm{KL}\left(q_\phi(\theta) \parallel p(\theta)\right),$$

which releases us from the intractable posterior and consists only of tractable distributions. This allows us to make direct usage of the vast gradient-based optimization literature to optimize the objective. The final form also gives us the typical form of an objective decomposing into a term controlling the data fit, here the expected log-likelihood of the data, and a second term serving as a regularizer, here the Kullback-Leibler divergence between the variational posterior and the prior.

The prior and variational posterior are both under our control, and can therefore be chosen so that this KL term is either analytically tractable or can be closely approximated. The first term has the problem that the expectation, which we could approximate via samples, is computed with respect to the density $q_\phi(\theta)$, which depends on the parameters $\phi$ we require the gradients of. There are two common approaches to solve this problem.

*Score gradient*

The first approach uses the fact that we can reformulate the gradient of the expectation as

$$\begin{aligned}
\nabla_\phi \mathbb{E}_{q_\phi(\theta)} \left[\log p(\mathcal{D}|\theta)\right] &= \nabla_\phi \int \log p(\mathcal{D}|\theta) q_\phi(\theta) d\phi \\
&= \int \log p(\mathcal{D}|\theta) \nabla_\phi q_\phi(\theta) d\theta \\
&= \int \log p(\mathcal{D}|\theta) \frac{\nabla_\phi q_\phi(\theta)}{q_\phi(\theta)} q_\phi(\theta) d\theta \\
&= \int \log p(\mathcal{D}|\theta) \nabla_\phi \log q_\phi(\theta) q_\phi(\theta) d\theta \\
&= \mathbb{E}_{q_\phi(\theta)} \left[\log p(\mathcal{D}|\theta) \nabla_\phi \log q_\phi(\theta)\right],
\end{aligned}$$

leaving the expecation to be approximated via samples. As this relies only on the score function $\nabla_\phi \log q_\phi(\theta)$, it is applicable to a large family of distributions without too many constraints. However, it tends to introduce a relatively large variance into the gradient.

*Reparametrization trick*

The second approach is known as the reparameterization trick (Kingma and Welling, 2014) and is applicable for distributions that allow for a reparametrization into a parameter-free base distribution together with a deterministic parametric transformation. For example using for a normal variational posterior with $\phi = (\mu, \sigma)$ we have that

$$\theta \sim \mathcal{N}(\theta|\mu, \sigma^2) \quad \Rightarrow \quad \theta = \mu + \sigma\varepsilon, \qquad \varepsilon \sim \mathcal{N}(\varepsilon|0,1),$$

which allows us to rewrite the expectation as

$$\mathbb{E}_{q_\phi(\theta)} \left[\log p(\mathcal{D}|\theta)\right] = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)} \left[\log p(\mathcal{D}|g(\phi,\varepsilon)\right].$$

While this tends to yield a lot less noisy gradients, it greatly limits the variety of applicable distributions. Throughout this work, we will rely on using normal distributions as the variational posteriors, which allows us to rely on this second variant which gives us a good enough gradient signal. Finally, we should mention that approaches to reduce this expectation term's gradient variance further, both in the score function as well as the reparametrization formulation, are still an ongoing research area (Roeder et al., 2017; Figurnov et al., 2018; Tucker et al., 2018), together with their extension to discrete variables (Jang et al., 2016; Maddison et al., 2016; Tucker et al., 2017).

Having learned such a variational posterior, we can then approximate the predictive distribution given a new data point $\boldsymbol{x}_*$ as

$$p(\boldsymbol{y}_*|\boldsymbol{x}_*, \mathcal{D}) = \int p(\boldsymbol{y}_*|\theta, \boldsymbol{x}_*) p(\theta|\mathcal{D}) \, \mathrm{d}\theta \tag{2.5}$$

$$\approx \int p(\boldsymbol{y}_*|\theta, \boldsymbol{x}_*) q_\phi(\theta) \, \mathrm{d}\theta. \tag{2.6}$$

Depending on the specific choice of the likelihood and the variational posterior this integral might be tractable, but in practice, especially in the setting of Bayesian neural networks, where the mapping is still highly nonlinear, it needs to be approximated further via samples giving us finally

$$p(\boldsymbol{y}_*|\boldsymbol{x}_*, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(\boldsymbol{y}_*|\theta^{(s)}, \boldsymbol{x}_*), \qquad \theta^{(s)} \sim q_\phi(\theta). \tag{2.7}$$

### 2.1.2 *Probability and Uncertainty*

What is probability? What is uncertainty? Is there true randomness? These questions have occupied philosophers and scientists throughout the centuries and will keep occupying them in the future. We will not provide a broad philosophical discussion here summarizing the different lines of thought and the many different schools and subschools, arguing for or against a specific interpretation. Instead, we summarize the interpretation used within this thesis and invite the reader to a cup of tea to meet for a philosophical discussion after he or she has finished the thesis. Throughout this work, we follow a Bayesian approach of using probability to quantify our uncertainty, by using it to encode our belief state about the world, conditioned on our past observations. How to update these beliefs given new information consistently is then prescribed by the Bayes rule.

In this sense, there is just one probability and thus, uncertainty. However, the machine learning literature distinguishes three different types of uncertainty, popularized in the deep learning literature by Kendall and Gal (2017). These are *aleatoric*, *epistemic*, as well as *predictive* uncertainty.

*aleatoric vs epistemic uncertainty*

For us epistemic uncertainty is the model uncertainty about our parameters, i.e. the uncertainty in their posteriors distributions after having observed some data. It is a reducible uncertainty, as further observations can help us learn

more about the relationship between the data and the parameters, becoming more certain about it. Aleatoric is the uncertainty that remains independent of the number of observations, i.e. is irreducible within our model. The condition "within our model" is important to note here as a different model might be able to reduce this uncertainty, blurring the line between the two somewhat. Predictive uncertainty lastly is the uncertainty in our predictions after marginalizing over all our parameters, giving us the posterior predictive, which combines every source of uncertainty.

## 2.2 (BAYESIAN) NEURAL NETS

This section does not aim to give a full introduction and overview of the world of neural networks and deep learning. See for example the recent textbook by Goodfellow et al. (2016) for that. Its aim is primarily to introduce the notation as we will use it in the following chapters and in the second part introduce the variant of Bayesian Neural networks that is the most common one. That way we combine the abstract discussion of variational inference from the last section with a specific use case, which will serve as an important baseline throughout the rest of this work.

One thing to note is that although we will later also rely on architectures with convolutional layers, the theoretical discussions here and in the following chapters always focus notationwise on the fully-connected use case. However, they usually directly generalize unless we explicitly discuss differences.

*Neural Network*

NEURAL NETWORK Throughout this work we will assume the following structure/architecture of the models we consider. The consist of

- an input $\boldsymbol{x} \in \boldsymbol{R}^{n_0}$, which we usually assume to be deterministic,
- $L$ layers with weights $\boldsymbol{W}_l \in \boldsymbol{R}^{n_l \times n_{l-1}}$ for $l = 1, \dots, L$,
- a nonlinear activation function $\mathrm{r}(\cdot)$ between them, which is applied elementwise to its input.

A neural net layer is then just a matrix-vector multiplication followed by the elementwise application of some nonlinear function. Refering to weight matrices collectively as $\theta = (W_1, \dots, W_L)$, and to the network as a function $f(\cdot; \theta)$, we have for example that a three layer neural net has the following structure

$$f(\boldsymbol{x}; \theta) = \boldsymbol{W}_3 r(\boldsymbol{W}_2 r(\boldsymbol{W}_1 \boldsymbol{x})),$$

where we incorporated the bias terms into the weight matrices, which we will continue to do, suppressing the bias terms from the notation.

As we will often be required to consider a net not in its totality, but on the level of the individual layers, we will refer to the output of the $l$-th layer before the subsequent application of an activation function as the *pre-activation* $\boldsymbol{f}_l$, and after the application of the activation function as the *post-activation* $\boldsymbol{h}_l$.

*pre-activation*
*post-activation*

Summarized as equations, this gives us

$$\begin{aligned} \boldsymbol{f}_l &\triangleq \boldsymbol{W}_l \boldsymbol{h}_{l-1} \\ \boldsymbol{h}_l &\triangleq r(\boldsymbol{f}_l), \end{aligned}$$

where we use $\boldsymbol{h}_0 = \boldsymbol{x}$ as a further shorthand and $r(\cdot)$ applies to each dimension individually, i.e. elementwise. The computation of each of these terms, starting from the input to the output layer is known as the *forward pass*. Given the dimensionalities specified above $\boldsymbol{h}_l$ is an $n_l$ dimensional vector, which to stay with the metaphor of the brain are referred to as $n_l$ neurons in the literature.

BAYESIAN NEURAL NETWORK Bayesian Neural Networks (BNNs) have been studied since at least the early nineties (MacKay, 1992, 1995; Neal, 1995; Lampinen and Vehtari, 2001). In parallel with the recent growth in popularity of deterministic neural nets, they have become popular again (Blundell et al., 2015; Kingma et al., 2015) and are an active field of study as we will see throughout this thesis. Like our discussion of their deterministic counterparts, the field is moving too fast for this to be the right place for a complete overview. Instead, we will introduce the two most common approaches, which we will also revisit throughout the next chapters as they will reappear as comparisons and baselines.

*Bayesian Neural Network*

The first of the two makes use of a common regularization strategy in the deterministic neural network literature. Dropout (Srivastava et al., 2016) consists of dropping a random set of neurons per layer, that is setting them to zero and ignoring their actual value, during each forward pass, each with a specific probability $p$. Introduced originally as a regularization technique to avoid overfitting Gal and Ghahramani (2016a,b) showed in a series of papers that this approach could be seen as implicitly performing variational inference with a specific family of variational distributions. This strategy is popular as it can trivially be applied to any existing architecture with neither theoretical nor real implementation constraints. We will refer to it as *MCDropout* throughout the rest of this work.

*MCDropout*

The second approach makes the connection to variational inference and a Bayesian approach even closer. The generative model is given as

$$\begin{aligned} \boldsymbol{W}_l &\sim p(\boldsymbol{W}_l), \qquad l = 1, \ldots, L \\ \boldsymbol{y}_n &\sim p(\boldsymbol{y}_n | \boldsymbol{x}_n, \theta), \qquad \forall n. \end{aligned}$$

Assuming a mean-field variational posterior over the weights $q(\theta)$ to be a multivariate normal distribution we have as the parameters to be learned the mean and variance paramters of each of the weights, summarized as $\phi$. The ELBO objective to be maximize is then as before

$$\mathbb{E}_{q_\phi(\theta)} \left[ \log p(\mathcal{D}|\theta) \right] + \mathrm{KL} \left( q_\phi(\theta) \parallel p(\theta) \right).$$

Given the usual choice of a normal prior, the second term is a Kullback-Leibler divergence between normal distributions, which is analytically tractable. We are not limited to this prior and could instead rely on sparsity inducing priors (Ghosh and Doshi-Velez, 2017), or more heavy-tailed distributions (Fortuin et al., 2020). We will however stick to a normal prior in the experiments.

*Reparametrization trick*

To make the expectation independent of the parameters to be optimized, we rely on the *reparametrization trick* (Kingma and Welling, 2014) already discussed in the general variational inference introduction and use the deterministic transformation of a parameter-free base distribution as

$$a \sim \mathcal{N}(a|\mu, \sigma^2) \Rightarrow a = \mu + \sigma\varepsilon, \qquad \varepsilon \sim \mathcal{N}(\varepsilon|0,1).$$

Denoting this transformation via $g(\cdot;\mu,\sigma)$ we have

$$\mathbb{E}_{q_\phi(\theta)}\left[\log p(\mathcal{D}|\theta)\right] = \mathbb{E}_{\mathcal{N}(\varepsilon|0,1)}\left[\log p(\mathcal{D}|g(\varepsilon;\phi)\right],$$

allowing us to approximate the resulting expectation with a simple Monte Carlo approximation, which during training is often further reduced to relying on a single sample as an approximation.

*Local reparametrization trick*

*VarOut*

Throughout the forward pass instead of sampling weights, we finally rely on the fact that a matrix product with a multivariate normal vector is again normally distributed and instead sample the output of each layer, referred to as the *local reparametrization trick*, which greatly reduces the gradient variance as Kingma et al. (2015) showed. As this means a sampling of the pre-activations, and is as such similar to the vanilla dropout we will refer to it as *Variational Dropout (VarOut)* in the experiments. It can be interpreted as replacing the Bernoulli dropout with a normal dropout where the dropout rate is not specified a priori but rather learned adaptively.

As the neural network setting usually involves a large number of data points, the log-likelihood can further be approximated when computing the gradients via a batch of size $M$ as

$$\log p(\mathcal{D}|\theta) = \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n,\theta) \approx \frac{N}{M}\sum_{m=1}^{M} \log p(\boldsymbol{y}_m|\boldsymbol{x}_m,\theta).$$

This introduction has solely relied on variational methods instead of purely sampling-based approaches to training Bayesian neural networks. Springenberg et al. (2016) for example have successfully relied on using stochastic gradient HMC (SGHMC) (Chen et al., 2014), a stochastic mini-batch approximation to Hamiltonian Monte Carlo (Betancourt, 2017) to train BNNs for the task of Bayesian optimization, where smaller networks are sufficient, but in general, such approaches greatly suffer from scalability issues which is why variational inference based approaches seem to be preferred. We will rely solely on the variational setup throughout this work.

*Regression and Classification*

REGRESSION AND CLASSIFICATION For most of the following chapters, we will consider regression and classification tasks, using a normal likelihood for the first case and a categorical for the second.

Assuming a homogeneous observation noise we have for a target $y$ in the case of regression that

$$y = f(\boldsymbol{x}; \boldsymbol{\theta}) + \varepsilon,$$

with $\varepsilon \sim \mathcal{N}(\varepsilon|0, \sigma^2)$. While for the classification case, assuming the task is to classify among $K$ categories in a *multi-class* setting, we have that

$$P(y = k) = \zeta(f(\boldsymbol{x}; \boldsymbol{\theta}))_k. \tag{2.8}$$

Here we have defined $\zeta(\cdot)$ as the *softmax function*. We will also consider the *multi-label* setup where $\boldsymbol{y} \in \{0, 1\}^K$ can potentially belong to multiple classes at the same time, in which case we have as the likelihood that

$$p(\boldsymbol{y}) = \prod_{k=1}^{K} \mathcal{B}er\big(\boldsymbol{y}_k|\sigma(f(\boldsymbol{x}; \theta)_k)\big), \tag{2.9}$$

and have $\sigma(\cdot)$ as the logistic sigmoid.

$$\zeta(\boldsymbol{a})_j \triangleq \frac{\exp(a_j)}{\sum_k \exp(a_k)}$$

$$\sigma(a) \triangleq \frac{1}{1+\exp(-a)}$$

## 2.3 ACTIVE LEARNING

As we will discuss the concept of *Active Learning (AL)* together with the adaptions we suggest in mathematical detail in Chapter 4, this section will remain on the abstract equation-free level.

Broadly speaking, we can consider active learning as belonging somewhere between supervised and unsupervised learning. The former has access to a pre-existing set of data pairs $(\boldsymbol{x}, y)$ and wants to find a relationship between them to predict the targets of newly arriving unlabeled data optimally. On the other hand, unsupervised learning algorithms only have access to $\boldsymbol{x}$ trying to find patterns within the observations without a specific prediction task.

Active learning is motivated by the realization that in practice, data becomes cheaper and cheaper. Simultaneously, the acquisition of the targets belonging to a specific $\boldsymbol{x}$ often remains expensive insofar as it requires human intervention and usually domain knowledge of the particular task to be solved. As such active learning still wants to learn a predictive relationship between features and target but tries to minimize the number of labelled data pairs required. It relies on the realization that not all data are created equal for each task. Instead of learning given a fixed training data set, the algorithm learns from a small initial data set and has access to an additional extensive unlabeled amount of data. Assigning a score to each of the unlabeled data points, based on how useful they likely are for improving the overall performance if a label were provided, the highest-scoring points can be provided to a human, an "oracle", to be labelled. The underlying model is then be retrained given the new information, and the process can be iterated until the desired performance is reached. See Figure 2.2 for a summary of the circular structure of this process.

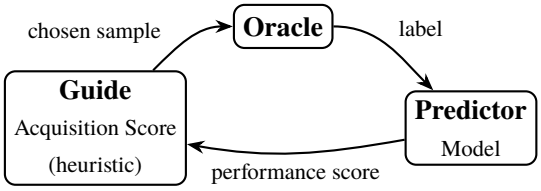



**Figure 2.2: Standard Active Learning pipeline.** The standard active learning pipeline is summarized as the interplay between three parts. An *oracle* provides a set of labeled data for a *predictor* to learn on. It, in turn, provides some signal to the *guide*, a usually fixed, hard-coded acquisition function, which communicates to the oracle which points to label next, restarting the cycle.

Metaphorically one can consider the example of a child learning. It is given some supervision and can explicitly request further information on objects or relationships it does not yet know or understand. This rough metaphor also indicates that active learning can be seen as part of a much larger lifelong learning framework. A model is not considered trained to convergence and done, but gets updated throughout its usage. Similarly to a child learning a lot in an unobserved interactive fashion, unsupervised learning on the unlabeled data and data augmentation on the labeled observations offers great potential, and is actively pursued in the literature.

We will in Chapter 4 focus solely on the plain setting of active learning, as it already poses a lot of open questions to be solved. For example what kind of a model can learn well and efficiently from little data, while also being useful in practice and additionally provide a helpful signal of how it could be improved to have an efficient way to request new labels. Of course, this in practice also becomes an even larger question, as the score should not only reflect how the model can be best improved and but also how to accommodate the human expert effectively. For example a model learning to perform image segmentation, i.e. a pixel-level classification of images, might be optimally improved by labelling individual pixels in separate images. Still, in practice, it will be more efficient to label parts of an image directly even though not all pixels might be strictly necessary.

See Settles (2012) for a recent textbook on active learning. Note also that this active learning task formulation is closely related to what is known as Bayesian Optimization (Snoek et al., 2012; Agnihotri and Batra, 2020). The task there is not to optimize the time/cost of a human expert by optimally requesting new labels. Rather the task can be seen as a meta-optimization step, where the goal is to optimize the performance of an underlying model for as little training time as possible. A second model, usually a Gaussian Process, is here trained to suggest new hyperparameters, that hopefully improve performance, to be evaluated next. Given the long training run time of current deep learning models and their sensitivity to hyperparameter choices, a simple grid search that iterates over all possible configurations is not even remotely possible (even if we were to ignore the fact that many hyperparameters are not discrete, but continuous).

Optimizing which set of parameters to evaluate next can then either rely on human expertise and clever decisions (see, e.g. Isensee et al. (2020) for an extensive study on how to set parameters in the task of image segmentation), or such a Bayesian optimization scheme. It was, for example employed with great success in the training of Alpha Go and similar models (Chen et al., 2018b; Silver et al., 2018).

## 2.4 REINFORCEMENT LEARNING VIA MARKOV DECISION PROCESSES

The field of *Reinforcement Learning (RL)* is huge and vastly growing, so this will not be an exhaustive overview. The primary source we base our introduction in this section on is the excellent textbook by Sutton and Barto (2018), which gives a complete introduction to the field. We will only focus on one concept from reinforcement learning that is most relevant for Chapter 4. This is the concept of a *Markov Decision Process (MDP)*.

*Markov Decision Process*

In an MDP, we consider for discrete time steps $t = 1, 2, \ldots$, an agent interacting with an environment. Given that the environment is in a state $S_t$ at time $t$, the agent performs an action $A_t$, which in turn influences the environment resulting in a state $S_{t+1}$ and a reward $R_{t+1}$. The model is fully specified by the set of possible actions $\mathcal{A}$ (which we consider to be discrete), a set of also discrete possible states $\mathcal{S}$, rewards $\mathcal{R} \subseteq \mathbb{R}$, and transition probabilities specifying the relationship between the time steps, as

*Transition probability*

$$p(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a),$$

i.e. the probability of observing a state $s'$ and receiving reward $r$, after the agent performs an action $a$ in state $s$. Given this tuple of specifications $(\mathcal{S}, \mathcal{R}, \mathcal{A}, p)$, the task is to learn how to perform the actions to maximize the cumulative reward. That is to maximize the expected future returns,

*return*

$$G_t \triangleq R_{t+1} + \cdots + R_T,$$

assuming a maximum of $T$ time steps. Or in the case of a setup without a fixed end point, we define it with a discount factor $\gamma \in [0, 1)$ as

$$G_t \triangleq \sum_{i=0}^{\infty} \gamma^i R_{t+1+i} = R_{t+1} + \gamma G_{t+1},$$

giving us a recursive relationship.

To optimize this we require two additional functions. The *policy* $\pi$ specifies how to act given a specific state, i.e. it gives the probability that $A_t = a$, given that $S_t = s$, which we denote as $\pi(a|s)$. The second is the *value function* $v_\pi(\cdot)$, giving the value of being in a specific state and following a particular policy $\pi$ as the expected return

*Policy*

*Value function*

$$v_\pi(s) \triangleq \mathbb{E}_\pi \left[ G_t | S_t = s \right].$$

It gives a similar recursive relationship, known as the *Bellman equation*

$$v_\pi(s) \triangleq \mathbb{E}_\pi \left[ G_t | S_t = s \right]$$

$$= \mathbb{E}_\pi \left[ R_{t+1} + \gamma G_{t+1} | S_t = s \right]$$
$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \big( r + \gamma v_\pi(s') \big).$$

A policy $\pi$ is then better as a policy $\pi'$ if its value function is higher, i.e. $v_\pi(s) \geq v_{\pi'}(s), \forall s$, giving rise to the *optimal value* as $v^*(s) \triangleq \max_\pi v_\pi(s)$ for a specific state $s$.

*Policy gradient and REINFORCE*

While there are a wide variety of methods how to solve this task, depending on the underlying structure of states and actions, we will focus the discussion here only on the *policy gradient* method as it will be the relevant approach we pursue later.

It considers the policy as a parametric function, which we denote here as $\pi_\theta(s, a)$ for some set of parameters $\theta$, optimizing them via a gradient ascent approach given some performance score

$$J(\theta) \triangleq v_{\pi_\theta}(s_0),$$

for some start state $s_0$. It can be shown (see e.g. Chapter 13 of Sutton and Barto (2018) for a proof), that its gradient is then given as

$$\nabla J(\theta) \propto \mathbb{E}_\pi \left[ G_t \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \right] = \mathbb{E}_\pi \left[ G_t \nabla \log \pi(A_t|S_t, \theta) \right],$$

known as the REINFORCE algorithm (Williams, 1992). This allows us to update the parameters of the policy given only an observed sequence of actions, states, and rewards, to improve the desired value function. We assume that the gradients of $\pi$ with respect to $\theta$ are available, which they usually are by design, e.g. in our case due to the representation of $\pi$ via a neural net.

Additionally it is common to introduce what is known as a *baseline* $b(\cdot)$ with an expected value of zero, but which still helps to reduce the gradient variance. This gives us the final gradient ascent updates as

$$\theta_{t+1} \leftarrow \theta_t + \lambda (G_t - b(S_t)) \nabla \log \pi(A_t|S_t, \theta_t),$$

for some learning rate $\lambda$. The baseline formulation here uses that for some density $p(x)$ and a constant $c$ we have

$$\mathbb{E}_{p(x)} \left[ c \nabla \log p(x) \right] = \int c p(x) \nabla \log p(x) dx$$
$$= c \int \nabla p(x) dx = c \nabla \int p(x) dx = 0,$$

which holds as long as $b(\cdot)$ is independent of the actions taken at time $t$.

## 2.5 PAC-BAYES

Chapters 5 and 6 make strong use of the concept of *Probably Approximately Correct (PAC)* learning, a term which dates back to Valiant (1984), but the

concept itself is well known in statistcs in general. While we rely in those chapters primarily on the PAC-Bayes theory (see, e.g. Guedj (2019) for a recent tutorial) to derive theoretically justified objectives to be optimized, and less for what it was initially designed for we will nevertheless give a high-level introduction to the concept here and get more concrete in those chapters.

The concept PAC refers to is: Given that we have trained our model on a set of examples, how much can we trust it? In the sense of how much does the performance as measured by some objective generalize to unseen data.

A PAC approach tries to address this by formalizing this desire stating that the *observed* performance should be close to the performance on *unseen* data, with a high probability, i.e.

$$\mathbb{P}\big(|\text{observed} - \text{unseen}| \leq \varepsilon\big) > 1 - \delta, \tag{2.10}$$

for some $\delta \in (0, 1]$.

To be mathematically more precise, assume that we have an observed data set $\mathcal{D}$ consisting of $N$ pairs $(\boldsymbol{x}_n, y_n)$ coming from an unobserved distribution $\Delta$, i.e. $\mathcal{D} \sim \Delta^N$. Additionally, we have a predictor $h(\cdot)$ that tries to predict the target $y_n$ given the input $\boldsymbol{x}_n$. Its performance is measured by some loss function $l(\cdot, \cdot)$, for example a squared difference in the case of regression. The observed performance is given by the *empirical risk* *Empirical risk*

$$R_{\mathcal{D}}(h) \triangleq \frac{1}{N} \sum_{n=1}^{N} l\big(h(\boldsymbol{x}_n), y_n\big).$$

Its counterpart is the unknown *theoretical risk* *Theoretical risk*

$$R(h) \triangleq \mathbb{E}_{\boldsymbol{x},y\sim\Delta}\left[l(h(\boldsymbol{x}), y)\right].$$

The goal is then to derive a generalization bound

$$R(h) \leq R_{\mathcal{D}}(h) + \varepsilon(N, \delta)$$

that holds with a probability of $1 - \delta$, for a suitable $\varepsilon(\cdot)$, similar to the construction of a $1 - \alpha$ confidence interval.

Take for example a single predictor and the task of classification with the loss being the error, such that

$$l(h(\boldsymbol{x}), y) \triangleq \begin{cases} 1, & \text{if } h(\boldsymbol{x}) = y \\ 0, & \text{else} \end{cases}.$$

Applying the Hoeffding inequality (see appendix) gives us *Hoeffding's inequality*

$$\mathbb{P}(|R(h) - R_{\mathcal{D}}(h)| > \varepsilon) \leq 2\exp(-2N\varepsilon^2),$$

and choosing $\varepsilon = \sqrt{\frac{\log(2/\delta)}{2N}}$ gives the desired probabilistic guarantee. For a finite set of predictors $\mathcal{H}$ this quarantee can directly be extended such that

$$\mathbb{P}(\forall h \in \mathcal{H} : |R(h) - R_{\mathcal{D}}(h)| > \varepsilon) \leq 2Me^{-2N\varepsilon^2},$$

assuming $|\mathcal{H}| = M$. A modified $\varepsilon = \sqrt{\frac{\log(2M/\delta)}{2N}}$ keeps the desired bound.

In order to extend this to infinite hypothesis spaces $\mathcal{H}$, the *Vapnik-Chernvonenkis (VC) dimension* can be used in order to derive a similar generalization bound (see e.g. Wasserman (2013)).

*PAC-Bayes*

The PAC-Bayesian approach (McAllester, 1998; Germain et al., 2016) varies from the discussion so far in that it introduces two specific distributions over the set $\mathcal{H}$. The first $P$, is any distribution over $\mathcal{H}$ independent of the observed data, and the second any data-dependent, learned, distribution $Q$, over the same space of hypothesis. In that sense, it is similar to the Bayesian setup, where we would have a prior and a posterior. However, it is more general. For the Bayesian framework, these two distributions are highly related insofar as they are coupled via the Bayes Theorem. PAC-Bayes generalizes this in that it allows us to consider any two distributions, as long as they fullfil the constraints of the former being independent of the data.[3]

This approach has given rise to a wide variety of bounds (e.g. McAllester, 1998, 1999; Catoni, 2007). One bound we will rely on during Chapter 6, relies on an adaptation of a bound due to Maurer (2004), which we summarize here in a slightly looser form which gives us a more explicit structure (Tolstikhin and Seldin, 2013).

*A PAC-Bayes bound*

A PAC-BAYES BOUND. *For any prior $P$ over some family of hypothesis $\mathcal{H}$, $N > 8$ observations, and any $\delta \in (0, 1]$ we have that*

$$\mathbb{P}\left(\forall Q \text{ on } \mathcal{H} : R(Q) \leq R_{\mathcal{D}}(Q) + \sqrt{\frac{\mathrm{KL}\left(Q \parallel P\right) + \log\frac{2\sqrt{N}}{\delta}}{2N}}\right) \geq 1 - \delta.$$

*where $KL\left(\cdot \parallel \cdot\right)$ speficies the Kullback-Leibler divergence.*

## 2.6 STOCHASTIC DIFFERENTIAL EQUATIONS

This overview on SDEs will, by necessity, remain at a relatively high level. As we will require them primarily for the chapter on learning SDEs, i.e. Chapter 6, we will discuss the relevant parts needed in that chapter here, and refer the reader to Särkkä and Solin (2019) for a recent textbook length treatment of the topic, that introduces the subject from the ground up, and that we also rely on for the following summary of results.

Consider first an arbitrary *Ordinary Differential Equation (ODE)*,

$$\frac{\mathrm{d}\boldsymbol{h}_t}{\mathrm{d}t} = f(\boldsymbol{h}_t, t), \tag{2.11}$$

with some known starting condition $\boldsymbol{h}_0$. We use the shorthand $\boldsymbol{h}_t \triangleq \boldsymbol{h}(t)$ to be consistent with later notation. Instead of considering analytical solutions that

3 $Q$ could potentially be also independent of the data, but then the whole task of learning and evaluating a generalization guarantee would become somewhat useless.

depend on the problem's specific structure, we can rely on numerical approximations to solve it. Integrating over a small time period $\Delta t$ we have

$$\boldsymbol{h}_{t+\Delta t} = \boldsymbol{h}_t + \int_t^{t+\Delta t} f(\boldsymbol{h}_\tau, \tau)\,\mathrm{d}\tau \approx \boldsymbol{h}_t + f(\boldsymbol{h}_t, t)\Delta t,$$

where the tightness of the approximation depends on the size of $\Delta t$. We can then solve the differential equation by what is known as the *Euler Method*.

*Euler Method*

EULER METHOD In order to approximately solve the differential equation (2.11) over a time interval $[0, T]$ (with $T > 0$) split the interval into $K$ parts $0 = t_0 < t_1 < \cdots < t_K = T$ and approximate the solution recursively as

$$\hat{\boldsymbol{h}}_{t_{k+1}} = \hat{\boldsymbol{h}}_{t_k} + f(\hat{\boldsymbol{h}}_{t_k}, t_k)\Delta t_k, \qquad k = 0, \ldots, K$$

where $\Delta t_k = t_{k+1} - t_k$.

The order of a numerical integration is defined as the largest $\rho$ such that there exists a constant $C$ for which the following inequality holds

$$|\hat{\boldsymbol{h}}_{t_K} - \boldsymbol{h}_{t_K}| \leq C\Delta t^\rho,$$

where we assumed $K$ equally spaced steps of length $\Delta t$. The Euler method, as defined above, can be shown to have order $\rho = 1$. In practice, there is a large family of better numerical integration methods. However, as we will rely on a variant of the Euler method for stochastic differential equations, we require only this approximation scheme.

*Stochastic Differential Equation*

Why extend the ODE setup from above to a *Stochastic Differential Equation (SDE)*? One of the benefit of the extension from ODEs to SDEs is that it allows us to model uncertainty in the task at hand, be it due to inherent randomness in the dynamics or due to imprecision and errors in the model assumptions. The general form of an SDE is given as

$$\frac{\mathrm{d}\boldsymbol{h}_t}{\mathrm{d}t} = f(\boldsymbol{h}_t, t) + G(\boldsymbol{h}_t, t)W_t. \tag{2.12}$$

Here $f(\cdot,\cdot)$ is known as the *drift function*, while $G(\cdot,\cdot)$ is matrix-valued *diffusion function*[4]. The stochastic noise process $W_t$ finally is some white noise, defined formally as a random function with the two properties

1. $W_s$, $W_t$ are independent if $s \neq t$,
2. the mapping $t \mapsto W_t$ is a Gaussian Process (Rasmussen and Williams, 2006) with zero mean function and a Dirac delta correlation kernel $k(s,t) = \delta(t-s)\boldsymbol{Q}$, where $\boldsymbol{Q}$ is the spectral density of the process.

While other stochastic processes are possible and used, we rely only on this Gaussian process formulation with the spectral density given as the identity matrix in this work.

In analogy to the Euler method for ODEs, one can derive a similar approximation scheme for SDEs, known as the *Euler-Maruyama (EM) method*.

4 Also known as the dispersion function, but we will rely on the former term.

*Euler-Maruyama Method*

EULER-MARUYAMA METHOD In order to approximately solve the stochastic differential equation (2.12) over a time interval $[0, T]$ (with $T > 0$), split the interval into $K$ parts $0 = t_0 < t_1 < \cdots < t_K = T$ and approximate the solution as follows. Draw $\hat{\boldsymbol{h}}_{t_0} \sim p(x_{t_0})$ and iterate over the following two steps from $k = 1, \ldots, K$:

i) Sample
$$\Delta\boldsymbol{\beta}_k \sim \mathcal{N}(\Delta\boldsymbol{\beta}_k | 0, \boldsymbol{Q}\Delta t_k),$$

ii) Compute
$$\hat{\boldsymbol{h}}_{t_{k+1}} = \hat{\boldsymbol{h}}_{t_k} + f(\hat{\boldsymbol{h}}_{t_k}, t_k)\Delta t_k + G(\hat{\boldsymbol{h}}_t, t)\Delta\boldsymbol{\beta}_k.$$

*Strong/Weak convergence*

Assuming the time steps $\Delta t$ to be equally spaced, we can derive as in the ODE case the order of convergence. For stochastic methods, we distinguish between two different types of convergence. The strong order of convergence considers the expectation of the absolute difference and requires the existence of a constant $C$ such that

$$\mathbb{E}\left[\left|\hat{\boldsymbol{h}}_{t_K} - \boldsymbol{h}_{t_K}\right|\right] \leq C\Delta t^{\gamma}.$$

The weak convergence on the other hand considers instead the absolute value of the expectations, i.e.

$$\left|\mathbb{E}[\hat{\boldsymbol{h}}_{t_K}] - \mathbb{E}\left[\boldsymbol{h}_{t_K}\right]\right| \leq C\Delta t^{\alpha}.$$

For the Euler-Maruyama methods as discussed above it can be shown that these exponents are given as $\gamma = \frac{1}{2}$ and $\alpha = 1$.

See the textbooks by Särkkä and Solin (2019) and Kloeden and Platen (2011) for proofs to the claims above and a principled derivation of the Euler-Maruyama method, which requires a proper discussion of the Itô calculus.

# 3

# SAMPLING-FREE VARIATIONAL INFERENCE OF BAYESIAN NEURAL NETWORKS BY VARIANCE BACKPROPAGATION

The advent of deep learning libraries (Abadi et al., 2015; The Theano Development Team, 2016; Paszke et al., 2019) has made fast prototyping of novel neural net architectures possible by writing short and straightforward high-level code. Their availability triggered an explosion of research output on application-specific neural net design, which allowed for fast improvement of predictive performance in almost all fields where machine learning is used. The next grand challenge is to solve mainstream machine learning tasks with more time-efficient, energy-efficient, and interpretable models that make predictions with attached uncertainty estimates. Industry-scale applications also require models that are robust to adversarial perturbations (Szegedy et al., 2014; Goodfellow et al., 2015).

The Bayesian modelling approach provides principled solutions to all of the challenges mentioned above. Bayesian Neural Networks (BNNs) (MacKay, 1992) lie at the intersection of deep learning and the Bayesian approach that learns the parameters of a machine learning model via posterior inference (MacKay, 1995; Neal, 1995). We can upgrade a deterministic net with an arbitrary architecture and loss function to a BNN simply by placing a prior distribution over its parameters, turning them into random variables.

Unfortunately, the non-linear activation functions at the layer outputs render direct methods to estimate the posterior distribution of BNN weights analytically intractable. A recently established technique for approximating this posterior is Stochastic Gradient Variational Bayes (Kingma and Welling, 2014), which suggests reparameterizing the variational distribution and then Monte Carlo integrating the intractable expected data fit part of the ELBO. However, sample noise for a cascade of random variables distorts the gradient signal, leading to unstable training. Improving the sampling procedure to reduce the variance of the gradient estimate is an active research topic. Recent advances in this vein include the local reparameterization trick (Kingma et al., 2015) and variance reparameterization (Molchanov et al., 2017; Neklyudov et al., 2017).

We here follow a second research direction of approaches (Hernández-Lobato and Adams, 2015; Kandemir, 2018; Wu et al., 2019) that avoid Monte Carlo sampling and the associated precautions required for variance reduction and present a novel BNN construction that makes variational inference possible with a closed-form ELBO. Without a substantial loss of generality, we restrict

This chapter is based on and extends Haußmann et al. (2020b).

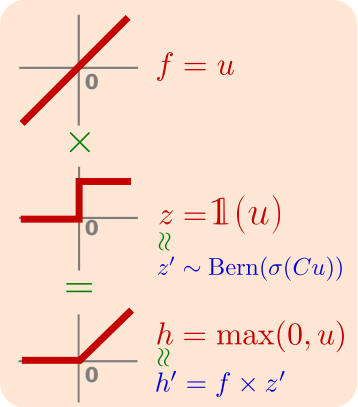



**Figure 3.1: ReLU decomposition.** We decompose the ReLU function into an identity function and a Heaviside step function, which is in turn approximated with a Bernoulli distribution.

the activation functions of all neurons of a net to the popular Rectified Linear Unit (ReLU) for most of this chapter and show how to extend it to other piecewise linear activation functions at the end. We build our formulation on the fact that the ReLU function can be expressed as the product of the identity function and the Heaviside step function: $\max(0, x) = x \cdot \mathbb{1}(x)$. The Heaviside step function is defined as

$$\mathbb{1}(x) \triangleq \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}.$$

Following Kandemir (2018) we exploit the fact that we are devising a probabilistic learner and introduce latent variables $z$ to mimic the deterministic Heaviside step functions. Throughout this chapter, we denote the distribution induced by this step function as

$$z \sim \delta_{x>0} \triangleq \mathcal{B}er(z|\mathbb{1}(x)),$$

i.e. a Bernoulli distribution that places either all of its mass on class one or class zero. This can then be relaxed to a non-deterministic Bernoulli

$$z \sim \delta_{x>0} \triangleq \mathcal{B}er(z|\mathbb{1}(x)) \approx \mathcal{B}er(z|\sigma(Cx)),$$

with some $C \gg 0$ and the logistic sigmoid function $\sigma(\cdot)$. The idea is illustrated in Figure 3.1.

We will show how the asymptotic account of this relaxation converts the likelihood calculation into a chain of linear matrix operations, giving way to a closed-form computation of the data-fit term of the Evidence Lower Bound in mean-field variational BNN inference. In our construction, the data-fit term lends itself as the sum of a standard loss (e.g. mean-squared error) on the expected prediction output and the predictor variance. This term has a recursive form, describing how the variance back-propagates through the layers of a BNN. We refer to our model as *Variance Back-Propagation* (VBP).

Experiments on several regression and classification tasks show that VBP can perform competitive to and improve upon other recent sampling-free as well as sampling-based approaches to BNN inference. Last but not least, VBP presents a generic formulation that is directly applicable to all weight prior selections as long as their Kullback-Leibler (KL) divergence with respect to the variational distribution is available in closed form, including the common log-Uniform (Kingma et al., 2015; Molchanov et al., 2017), Normal (Hernández-Lobato and Adams, 2015), and horseshoe (Louizos et al., 2017) priors.

## 3.1 BAYESIAN NEURAL NETS WITH DECOMPOSED FEATURE MAPS

Given a data set $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)_{n=1}^N\}$ consisting of $N$ pairs of $d$-dimensional feature vectors $\boldsymbol{x}_n$ and targets $y_n$, the task is to learn in a regression setting[1] with a normal likelihood

$$\begin{aligned} \boldsymbol{w} &\sim p(\boldsymbol{w}), \\ \boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w} &\sim \mathcal{N}\big(\boldsymbol{y}|f(\boldsymbol{X};\boldsymbol{w}), \beta^{-1}\mathbb{1}\big), \end{aligned}$$

for $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ and $\boldsymbol{y} = \{y_1, ..., y_N\}$, with $\beta$ as the observation precision, and $\mathbb{1}$ an identity matrix of suitable size[2]. The function $f(\,\cdot\,;\boldsymbol{w})$ specifies a feed-forward multi-layer neural net parameterized by weights $\boldsymbol{w}$, with ReLU activations between the hidden layers. $p(\boldsymbol{w})$ is an arbitrary prior over these network weights.

### 3.1.1 *The Identity-Heaviside Decomposition*

As mentioned in the introduction, a ReLU function can be decomposed into $\max(0, u) = u \cdot \mathbb{1}(u)$ This allows us to express the post-activation feature map vector $\boldsymbol{h}^l$ of a data point at layer $l+1$ as

$$\begin{aligned} \boldsymbol{h}^l &= \boldsymbol{f}^l \circ \boldsymbol{z}^l \\ \text{with } \boldsymbol{f}^l &= \boldsymbol{W}_l \boldsymbol{h}^{l-1} \text{ and } \boldsymbol{z}^l = \mathbb{1}\big(\boldsymbol{W}_l \boldsymbol{h}^{l-1}\big), \end{aligned}$$

where $\boldsymbol{h}^{l-1}$ is the feature map vector of the same data point at layer $l-1$, the matrix $\boldsymbol{W}_l$ contains the weights to map from layers $l-1$ to $l$ and $\circ$ denotes the element-wise Hadamard product of equal sized matrices or vectors. For $l = 0$, i.e. the input, we set $\boldsymbol{h}^0 = \boldsymbol{x}$. $\boldsymbol{f}^l$ is the linear pre-activation output vector of layer $l$. We refer to $\boldsymbol{z}^l$ as the activation vector. When the argument of the $\mathbb{1}(\cdot)$ function takes a vector as its input, we mean its elementwise application to all inputs. We denote the this factorized formulation of the feature map $\boldsymbol{h}^l$ as the *Identity-Heaviside Decomposition*.

1 See Section 3.1.7 for the extension to classification.

2 This formulation overloads the notation with $\mathbb{1}$ referring to a matrix, and $\mathbb{1}(\cdot)$ to the step-function. However, it should always be clear from the context which of the two is referred to.

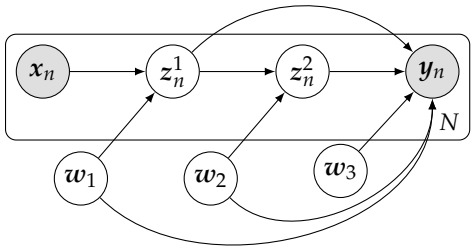



**Figure 3.2:** Plate diagram for a BNN with two hidden layers using the Identity-Heaviside decomposition.

Applying this decomposed expression for example on a feed-forward neural net with two hidden layers, we get[3]

$$y = f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}_3^\top \left( \boldsymbol{z}^2 \circ \boldsymbol{W}_2 (\boldsymbol{z}^1 \circ \boldsymbol{W}_1 \boldsymbol{x}) \right) + \varepsilon,$$

with $\varepsilon \sim \mathcal{N}(0, \beta^{-1})$, for a single data point consisting of the input-output pair $(\boldsymbol{x}, y)$. Note that given the binary activations $\boldsymbol{z}^1$ and $\boldsymbol{z}^2$ of the step function, the predictor output can be computed following a chain of *linear* operations.

### 3.1.2 *The Probabilistic Model*

For $z_{nj}^l$—the $j$th activation at the $l$th layer for data point $n$—we can approximate the Heaviside step function, by assuming that $z_{nj}^l$ is Bernoulli distributed

$$z_{nj}^l \sim \mathcal{B}er\left( z_{nj}^l | \sigma(C \cdot \boldsymbol{w}_{lj}^\top \boldsymbol{h}_n^l) \right), \tag{3.1}$$

$\sigma(x) \triangleq \frac{1}{1+\exp(-x)}$

where $\sigma(\cdot)$ is the logistic sigmoid function. The approximation becomes precise as $C \to \infty$.

Applying the Identity-Heaviside decomposition and Bernoulli relaxation to an $L$-layer BNN we obtain as the overall generative model

$$\begin{aligned} \boldsymbol{w}_{lj} &\sim p(\boldsymbol{w}_{lj}), \qquad \forall l, j \\ z_{nj}^l | \boldsymbol{w}_{lj}, \boldsymbol{h}_n^l &\sim \mathcal{B}er\left( z_{nj}^l | \sigma(C \cdot \boldsymbol{w}_{lj}^\top \boldsymbol{h}_n^l) \right), \quad \forall l, n, j \\ \boldsymbol{y} | \boldsymbol{w}, \boldsymbol{X}, \boldsymbol{Z} &\sim \mathcal{N}\left( \boldsymbol{y} | f(\boldsymbol{X}; \boldsymbol{w}), \beta^{-1} \mathbb{1} \right), \end{aligned}$$

where $\boldsymbol{Z}$ is the collection of activation variables $z_{nj}^l$. See Figure 3.2 for the plate diagram of a BNN with two hidden layers with this generative model.

#### 3.1.2.1 *Relation to V-ReLU-Net*

The decomposition of the ReLU activation has been proposed before in the context of learning BNNs as the Variational ReLU Network (V-ReLU-Net) (Kandemir, 2018). That work aimed to derive a gradient-free closed-form variational

3 We suppress the bias terms of each layer from the notation as it can directly be incorporated by extending the respective weights matrices.

inference scheme updating each variable to the local optimum conditioned on the other variables. This necessitates not only the decomposition we described above but also a further mean-field decoupling of the BNN layers. Translated to our notation, the V-ReLU-Net has for $h^l_{nj}$—the post-activation unit $j$ of layer $l$ and data point $n$—the following model

$$\begin{aligned}
\boldsymbol{w}_{lj} &\sim \mathcal{N}(\boldsymbol{w}_{lj}|0, \alpha^{-1}\mathbb{1}),\\
f^l_{nj}|\boldsymbol{w}_{lj}\boldsymbol{h}^{l-1}_n &\sim \mathcal{N}(f^l_{nj}|\boldsymbol{w}^\top_{lj}\boldsymbol{h}^{l-1}_n, \beta^{-1}),\\
z^l_{nj}|f^l_{nj} &\sim \mathcal{B}er(z^l_{nj}|\sigma(Cf^l_{nj})),\\
h^l_{nj}|z^l_{nj}, f^l_{nj} &\sim \mathcal{N}(h^l_{nj}|z^l_{nj}f^l_{nj}, \gamma^{-1}),
\end{aligned}$$

where $\alpha, \beta, \gamma$ are fixed hyperparameters. This means introducing additional normal distributions over each pre- & post-activation unit of each layer in the network. This setup allows one to learn a mean-field variational posterior, by iterating through the latent variables and updating the distribution of each to the optimal distribution conditioned on the others, thereby sidestepping the necessity of a gradient-based update scheme.

While such factorization across layers enjoys gradient-free variational update rules, it suffers from poor local maxima due to lack of direct feedback across non-neighbouring layers. Our formulation loosens this update scheme in the way we discuss in the following sections. Our experiments show that the benefit of gradient-free closed-form updates tends not to be worth the added constraints placed on the BNN for our kind of problems.

### 3.1.3 *Variational Inference of the Posterior*

In the Bayesian context, as discussed in the background chapter, learning consists of inferring the posterior distribution over the free parameters of the model

$$p(\theta|\mathcal{D}) = \frac{p(\boldsymbol{y}|\theta, \boldsymbol{X})p(\theta)}{\int p(\boldsymbol{y}|\theta, \boldsymbol{X})p(\theta)\,\mathrm{d}\theta}, \tag{3.2}$$

which is intractable for neural nets due to the integral in the denominator. Hence we need to resort to approximations. Our study focuses on variational inference due to its computational efficiency. It approximates the true posterior by a proxy distribution $q_\phi(\theta)$ with a known functional form parameterized by $\phi$ and minimizes the Kullback-Leibler divergence between $q_\phi(\theta)$ and $p(\theta|\mathcal{D})$

$$\mathrm{KL}\left(q_\phi(\theta) \parallel p(\theta|\mathcal{D})\right).$$

After a few algebraic manipulations, minimizing this KL divergence and maximizing the functional below turn out to be equivalent problems

$$\mathcal{L}_{\text{elbo}} = \underbrace{\mathbb{E}_{q_\phi(\theta)}\left[\log p(\boldsymbol{y}|\theta, \boldsymbol{X})\right]}_{\mathcal{L}_{\text{data}}} - \underbrace{\mathrm{KL}\left(q_\phi(\theta) \parallel p(\theta)\right)}_{\mathcal{L}_{\text{reg}}}. \tag{3.3}$$

This functional is often referred to as the *Evidence Lower BOund (ELBO)*, as it is a lower bound to the log marginal distribution $\log p(\boldsymbol{y}|\boldsymbol{X})$ known as the

evidence. The ELBO has the intuitive interpretation that $\mathcal{L}_{\text{data}}$ is responsible for the data fit, as it maximizes the expected log-likelihood of the data, and $\mathcal{L}_{\text{reg}}$ serves as a complexity regularizer by punishing unnecessary divergence of the approximate posterior from the prior.

In our setup, $\theta$ consists of the global weights $\boldsymbol{w}$ and the local activations $\boldsymbol{Z}$. For the weights, we follow prior art (Hernández-Lobato and Adams, 2015; Kingma et al., 2015; Molchanov et al., 2017) and adopt the mean-field assumption that the variational distribution factorizes across them. We assume each variational weight distribution to follow $q(w_{ij}^l) \triangleq \mathcal{N}(w_{ij}^l|\mu_{ij}^l, (\sigma_{ij}^l)^2)$ and parameterize the individual variances via their logarithms to avoid the positivity constraint giving us $\phi \triangleq \{(\mu_{ij}^l, \log \sigma_{ij}^l)_{ijl}\}$. We also assign an individual factor to each $z_{nj}^l$ local variable. Rather than handcrafting this factor's functional form, we calculate its ideal form having other factors fixed, as detailed below in Section 3.1.5. The final variational distribution is given as

$$q(\boldsymbol{Z})q_\phi(\boldsymbol{W}) \triangleq \prod_{n=1}^{N}\prod_{l=1}^{L}\prod_{i=1}^{n_{l-1}}\prod_{j=1}^{n_l} q(z_{nj}^l)q_\phi(w_{ij}^l), \tag{3.4}$$

where $n_l$ denotes the number of units at layer $l$.

This allows us to rewrite the ELBO $\mathcal{L}_{\text{elbo}}$ as

$$\begin{aligned}\mathcal{L}_{\text{elbo}} = \mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[\log p(\boldsymbol{y}|\boldsymbol{W},\boldsymbol{Z},\boldsymbol{X})\right] \\ - \mathbb{E}_{q_\phi(\boldsymbol{W})}\left[\mathrm{KL}\left(q(\boldsymbol{Z}) \parallel p(\boldsymbol{Z}|\boldsymbol{W},\boldsymbol{X})\right)\right] \\ - \mathrm{KL}\left(q_\phi(\boldsymbol{W}) \parallel p(\boldsymbol{W})\right),\end{aligned}$$

splitting it into three terms. As our ultimate goal is to obtain the ELBO in closed form, we have that for the third term any prior on weights that lends itself to an analytical solution of $\mathrm{KL}\left(q(\boldsymbol{W}) \parallel p(\boldsymbol{W})\right)$ is acceptable. We have a list of attractive and well-settled possibilities to choose from, including for example: i) the Normal prior (Blundell et al., 2015) for mere model selection, ii) the log-Uniform prior (Kingma et al., 2015; Molchanov et al., 2017) for atomic sparsity induction and aggressive synaptic connection pruning, and iii) the horseshoe prior (Louizos et al., 2017) for group sparsity induction and neuron-level pruning. Here, we stick to a simple normal prior to be maximally comparable to our baselines, rendering this term tractable.

*Decomposition of the ELBO*

We will discuss the second term of $\mathcal{L}_{\text{elbo}}$, $\mathbb{E}_{q_\phi(\boldsymbol{W})}\left[\mathrm{KL}\left(q(\boldsymbol{Z}) \parallel p(\boldsymbol{Z}|\boldsymbol{W},\boldsymbol{X})\right)\right]$ in greater detail in Proposition 2, which leaves the first term that is responsible for the data fit. For our regression likelihood, we can decompose $\mathcal{L}_{\text{data}}$ as

$$\begin{aligned}\mathcal{L}_{\text{data}} &= \mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[\log p(\boldsymbol{y}|\boldsymbol{W},\boldsymbol{Z},\boldsymbol{X})\right] \\ &\overset{c}{=} -\frac{\beta}{2}\sum_{n=1}^{N}\mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[\left(y_n - f(\boldsymbol{x}_n;\boldsymbol{w})\right)^2\right] \\ &= -\frac{\beta}{2}\sum_{n=1}^{N}\Big\{\left(y_n - \mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[f(\boldsymbol{x}_n;\boldsymbol{w})\right]\right)^2\end{aligned}$$

$$+ \operatorname{var}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[f(x_n;\boldsymbol{w})\right]\Big\}, \tag{3.5}$$

where $\stackrel{c}{=}$ indicates equality up to an additive constant, and the second equality is due to the common reformulation used in the bias-variance decomposition of the mean-squared error.

In this form, the first term is the squared error evaluated at the mean of the predictor $f(\cdot)$. The second term is its variance, which infers the total amount of model variance to account for the epistemic uncertainty in the learning task (Kendall and Gal, 2017). The desire for a sampling-free solution to (3.5) therefore translates to the requirement of an analytical solution to the expectation and variance terms.

*Posterior predictive in the case of regression*

POSTERIOR PREDICTIVE Before we derive how to compute the variance and expectation terms in a sampling-free manner, we first discuss how to perform prediction in the model. The posterior predictive for a test point $\boldsymbol{x}_*$

$$p(y_*|\boldsymbol{x}_*,\mathcal{D}) = \iint p(y_*|\boldsymbol{x}_*,\boldsymbol{W},\boldsymbol{Z})p(\boldsymbol{Z},\boldsymbol{W}|\mathcal{D})\,\mathrm{d}\boldsymbol{Z}\,\mathrm{d}\boldsymbol{W},$$

remains intractable, even using the proposed Identity-Heaviside decomposition. We therefore approximate it as

$$\begin{aligned} p(y_*|\boldsymbol{x}_*,\mathcal{D}) &\approx \iint \mathcal{N}(y_*|f_*^L,\beta^{-1})\mathcal{N}\left(f_*^L|\mathbb{E}\left[f_*^L\right],\operatorname{var}\left[f_*^L\right]\right)\mathrm{d}f_*^L \\ &= \mathcal{N}\left(y_*|\mathbb{E}\left[f_*^L\right],\beta^{-1}+\operatorname{var}\left[f_*^L\right]\right), \end{aligned}$$

where $f_*^L \triangleq f(\boldsymbol{x}_*;\boldsymbol{w})$. Assuming that we can compute these two terms, the posterior predictive can be approximately computed in closed form. Next, we discuss how to compute them in a sampling-free manner.

### 3.1.4 *Closed-form Calculation of the Data Fit Term*

*Computing the Expectation*

THE EXPECTATION TERM When all feature maps of the predictor are Identity-Heaviside decomposed and the $z_{nj}^l$'s are approximated by Bernoulli distributions as described in Section 3.1.2, the expectation

$$\mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[f(\boldsymbol{x}_n;\boldsymbol{w})\right]$$

can be calculated in closed form, as $f(\boldsymbol{x}_n;\boldsymbol{w})$ consists only of linear operations over independent variables according to our mean-field variational posterior assumption allowing the expectation to commute operation orders. This order interchangeability enables us to compute the expectation term in a single forward pass where each weight takes its mean value with respect to its related factor in the approximate distribution $q(\boldsymbol{Z})q_\phi(\boldsymbol{W})$. For instance, for our earlier example of a Bayesian neural net with two hidden layers, we have

$$\begin{aligned} &\mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[f(\boldsymbol{x}_n;\boldsymbol{w})\right] \\ &\quad= \mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[\boldsymbol{w}_3{}^\top(\boldsymbol{z}_n^2 \circ \boldsymbol{W}_2(\boldsymbol{z}_n^1 \circ \boldsymbol{W}_1\boldsymbol{x}_n))\right] \end{aligned}$$

$$= \mathbb{E}\left[\boldsymbol{w}_3\right]^\top \left(\mathbb{E}\left[\boldsymbol{z}_n^2\right] \circ \mathbb{E}\left[\boldsymbol{W}_2\right] \left(\mathbb{E}\left[\boldsymbol{z}_n^1\right] \circ \mathbb{E}\left[\boldsymbol{W}_1\right] \boldsymbol{x}_n\right)\right).$$

Consequently, we can calculate the squared error part of the data-fit term in closed form. This interchangeability property of linear operations against expectations holds as long as we keep independence between the layers. Hence one could also extend it to a non-mean-field case, where the weights of a single layer follow for example a general multivariate normal distribution without being constrained to a diagonal covariance matrix.

*Computing the Variance*

THE VARIANCE TERM CALCULATED VIA RECURSION The second term in (3.5) that requires an analytical solution is the variance

$$\mathrm{var}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[f(\boldsymbol{x}_n; \boldsymbol{w})\right].$$

Its derivation relies on using the following two identities on the relationship between the variances of two independent random variables $a$ and $b$:

$$\mathrm{var}\left[a + b\right] = \mathrm{var}\left[a\right] + \mathrm{var}\left[b\right], \tag{3.6}$$

$$\begin{aligned}\mathrm{var}\left[a \cdot b\right] &= \mathrm{var}\left[a\right]\mathrm{var}\left[b\right] + \mathbb{E}\left[a\right]^2 \mathrm{var}\left[b\right] + \mathrm{var}\left[a\right]\mathbb{E}\left[b\right]^2 \\ &= \mathbb{E}\left[a^2\right]\mathrm{var}\left[b\right] + \mathrm{var}\left[a\right]\mathbb{E}\left[b\right]^2. \end{aligned} \tag{3.7}$$

Applying these well-known identities to the linear output layer activations $\boldsymbol{f}^L$ of the $n$-th data point[4] we have

$$\begin{aligned}\mathrm{var}\left[\boldsymbol{f}^L\right] &= \mathrm{var}\left[\boldsymbol{w}_L^\top \boldsymbol{h}^{L-1}\right] = \mathrm{var}\left[\sum_{j=1}^{n_L} w_{Lj} h_j^{L-1}\right] \\ &= \sum_{j=1}^{n_L} \mathbb{E}\left[w_{Lj}^2\right]\mathrm{var}\left[h_j^{L-1}\right] + \mathrm{var}\left[w_{Lj}\right]\mathbb{E}\left[h_j^{L-1}\right]^2.\end{aligned}$$

Given the normal variational posterior over the weights, we directly have

$$\mathbb{E}\left[w_{Lj}^2\right] = \mu_{Lj}^2 + \sigma_{Lj}^2 \quad \text{and} \quad \mathrm{var}\left[w_{Lj}\right] = \sigma_{Lj}^2,$$

while $\mathbb{E}\left[h_j^{L-1}\right]$ consists again of a series of linear operations and can be computed as described above in the case of $\mathbb{E}\left[f(\boldsymbol{x}; \boldsymbol{w})\right]$. For $\mathrm{var}\left[h_j^{L-1}\right]$ finally we can use the second variance identity again and arrive at

$$\begin{aligned}\mathrm{var}\left[h_j^{L-1}\right] &= \mathrm{var}\left[z_j^{L-1} \cdot f_j^{L-1}\right] \\ &= \mathbb{E}\left[(z_j^{L-1})^2\right]\mathrm{var}\left[f_j^{L-1}\right] + \mathrm{var}\left[z_j^{L-1}\right]\mathbb{E}\left[f_j^{L-1}\right]^2.\end{aligned}$$

Combining these results we have for the $j$-th dimension of the post-activation $h_j^l$ of an arbitrary hidden layer $l$

$$\mathrm{var}\left[h_j^l\right] = \mathrm{var}\left[\sum_{i=1}^{n_l} z_i^l w_{ij}^l h_i^{l-1}\right]$$

4 We suppress the $n$ index throughout following derivations to reduce the notational overhead.

$$
\begin{aligned}
&= \sum_{i=1}^{n_l} \operatorname{var}\left[z_i^l w_{ij}^l h_i^{l-1}\right] \\
&= \sum_{i=1}^{n_l} \mathbb{E}\left[(z_i^l)^2\right] \operatorname{var}\left[w_{ij}^l h_i^{l-1}\right] + \operatorname{var}\left[z_i^l\right] \left(\mathbb{E}\left[w_{ij}^l\right] \mathbb{E}\left[h_j^{l-1}\right]\right)^2 \\
&= \sum_{i=1}^{n_l} \mathbb{E}\left[(z_i^l)^2\right] \left\{\mathbb{E}\left[(w_{ij}^l)^2\right] \operatorname{var}\left[h_i^{l-1}\right] + \operatorname{var}\left[w_{ij}^l\right] \mathbb{E}\left[h_i^{l-1}\right]^2\right\} \\
&\qquad + \operatorname{var}\left[z_i^l\right] \left(\mathbb{E}\left[w_{ij}^l\right] \mathbb{E}\left[h_j^{l-1}\right]\right)^2. \qquad (3.8)
\end{aligned}
$$

If we assume that we can evaluate $\mathbb{E}\left[(z_i^l)^2\right]$ and $\operatorname{var}\left[z_i^l\right]$, which we derive in the next section, the only term left to evaluate is the variance of the post-activations at the previous layer $\operatorname{var}\left[h_i^{l-1}\right]$. Hence, we arrive at a recursive description of the model variance. Following this formula, we can express $\operatorname{var}\left[f(x_n; \boldsymbol{w})\right]$ as a function of $\operatorname{var}\left[\boldsymbol{h}_n^{L-1}\right]$, then $\operatorname{var}\left[\boldsymbol{h}_n^{L-1}\right]$ as a function of $\operatorname{var}\left[\boldsymbol{h}_n^{L-2}\right]$, and repeat this procedure until the observed input layer, where variance is zero $\operatorname{var}\left[\boldsymbol{h}_n^0\right] = \operatorname{var}\left[\boldsymbol{x}_n\right] = 0$.

For noisy or otherwise probabilistic input, one can directly inject the desired variance model of the input data into the input layer, which would still not break the recursion and keep the formula valid, for example by assuming that the input features $\boldsymbol{x}_n$ are known up to an observation variance of $\operatorname{var}\left[\boldsymbol{x}_n\right] = \sigma^2$. Computing this variance term thus only requires a second pass through the network in addition to the one needed when the expectation term is computed, sharing many of the required calculations, allowing us to perform both efficiently. As this formula reveals how the analytical variance computation recursively back-propagates through the layers, we refer to our construction as *Variance Back-Propagation* (VBP).

Note that this section's derivations only consider an elementwise, diagonal, variance ignoring covariance terms between, e.g. $h_i^l$ and $h_j^l$. This is due to computational constraints, as the current approach allows us to pass two $n_l$ dimensional vectors, representing mean and variance forward to the next layer, while an explicit covariance construction would entail the creation of an $n_l \times n_l$ dimensional matrix for each layer, greatly increasing the computational cost. Our empirical evaluations show that this restriction does not cost us with respect to predictive performance. The central limit and moment matching based approaches we will discuss later on suffer from similar constraints and need to perform the same restrictions. For completeness, we will discuss this structure of the covariance and how to compute it in greater detail at the end of this chapter.

Learning the parameters $\phi$ of the variational posterior $q_\phi(\boldsymbol{W})$ to maximize this analytical form of the ELBO then follows via mini-batch stochastic gradient descent, using any of the common gradient update schemes.

### 3.1.5 *Updating the Binary Activations*

The results so far are analytically tractable contingent upon having the existence of a tractable expression for each of the $q(z_{nj}^l)$ factors in the variational posterior in (3.4). While we update the variational parameters of the weight factors via gradient descent of the ELBO, for the binary activation distributions $q(z_{nj}^l)$, we choose to perform the update at the function level. Benefiting from variational calculus, we fix all other factors in $q(\boldsymbol{Z})q_\phi(\boldsymbol{W})$ except for a single $q(z_{nj}^l)$ and find the optimal functional form for this remaining factor. We devise in Propositon 1 an approach for calculating such variational updates.

*Variational posterior of z*

PROPOSITION 1. *Consider a Bayesian model including the generative process excerpt below*

$$\begin{aligned} &\vdots \\ a &\sim p(a), \\ z|a &\sim \delta_{a>0} \approx \mathcal{B}er\big(z|\sigma(Ca)\big), \\ b|z,a &\sim p(b|g(z,a)), \\ &\vdots \end{aligned}$$

*for some arbitrary function $g(z,a)$ and $C \gg 0$. If the variational inference of this generative model is to be performed with an approximate distribution $Q = \cdots q(a)q(z)q(b)\cdots$[5], the optimal closed-form update for $z$ is*

$$q(z) \leftarrow \mathcal{B}er\big(z|\sigma(C\mathbb{E}_{q(a)}[a])\big),$$

*and for $C \to \infty$ we have*

$$q(z) \leftarrow \delta_{\mathbb{E}_{q(a)}[a]>0}.$$

For our specific case, this translates for a finite $C$ to

$$q(z_{nj}^l) \leftarrow \mathcal{B}er\Big(z_{nj}^l|\sigma\big(C\cdot\textstyle\sum_i \mathbb{E}\left[w_{ij}^l\right]\mathbb{E}\left[h_{ni}^{l-1}\right]\big)\Big),$$

and in the limit to[6]

$$q(z_{nj}^l) \leftarrow \delta_{\mathbb{E}\left[\boldsymbol{w}_j^l\right]^\top \mathbb{E}\left[\boldsymbol{h}_n^{l-1}\right]>0},$$

involving only terms that are already computed during the forward pass computation of the expectation term and can be done concurrently to that forward pass. The Bernoulli distribution in turn provides us analytical expressions for the remaining expectation and variance terms in the computation of (3.8).

5 Note that $q(b)$ might or might not exist depending on whether $b$ is latent or observed. We here assume that $b$ is latent, with the proof following analogously for an observed $b$.

6 Note that as this delta distribution is just a deterministic Bernoulli, its expectation is the binary outcome of the condition it tests and the variance is zero.

PROOF OF PROPOSITION 1. Consider the following reformulation of the Bernoulli probability mass function for a logistic sigmoid $\sigma(\cdot)$

$$\mathcal{B}er(z|\sigma(a)) = \sigma(a)^z(1-\sigma(a))^{1-z} = e^{az}\sigma(-a).$$

Using this reformulation and the general form of the optimal solution of $q(z)$ given the other parameters (see the Background chapter for details) the update can be computed as

$$\begin{aligned}
\log q(z) &\leftarrow \mathbb{E}_{q(a)q(b)}\left[\log p(a)p(z|a)p(b|z,a)\right] + \text{const} \\
&\overset{c}{=} \mathbb{E}_{q(a)}\left[\log p(a)\right] + \mathbb{E}_{q(a)}\left[\log p(z|a)\right] \\
&\quad + \mathbb{E}_{q(a)q(b)}\left[\log p(b|g(z,a))\right] \\
&\overset{c}{=} \mathbb{E}_{q(a)}\left[\log p(a)\right] + \mathbb{E}_{q(a)}\left[\log \sigma(-Ca)\right] \\
&\quad + Cz\mathbb{E}_{q(a)}\left[a\right] + \mathbb{E}_{q(a)q(b)}\left[\log p(b|g(z,a))\right].
\end{aligned}$$

Here, the first two terms do not depend on $z$ and can hence be dropped into the constant while of the third and fourth terms, the third will dominate for $C \gg 0$. We are then left with with the unnormalized logarithm of a Bernoulli density giving us

$$q(z) \leftarrow \mathcal{B}er\big(z|\sigma(C\mathbb{E}_{q(a)}\left[a\right])\big).$$

For $C \to \infty$, we get the desired

$$\lim_{C\to\infty} \sigma(C\mathbb{E}_{q(a)}\left[a\right]) = \delta_{\mathbb{E}_{q(a)}[a]>0}$$

□

#### 3.1.5.1 *The Expected KL Term on Z*

A side benefit of these optimal $q(z^l_{nj})$ distributions is that the complicated $\mathbb{E}_{q_\phi(W)}\left[\mathrm{KL}\left(q(\boldsymbol{Z}) \parallel p(\boldsymbol{Z}|\boldsymbol{W},\boldsymbol{X})\right)\right]$ term can be calculated analytically subject to a controllable degree of relaxation as we now show in Proposition 2.

PROPOSITION 2. *For the model and the inference scheme as discussed in Proposition 1 with $q(a) = \mathcal{N}(a|\mu,\sigma^2)$, in the relaxed delta formulation $\delta_{a>0} \approx \mathcal{B}er\big(a|\sigma(Ca)\big)$ with some finite $C > 0$, the expected KL divergence $\mathbb{E}_{q(a)}\left[\mathrm{KL}\left(q(z) \parallel p(z|a)\right)\right]$ is (i) approximately analytically tractable and (ii) its magnitude goes to $0$ quickly as $|\mu|$ increases, with $\sigma$ controlling how fast it drops towards $0$.*

PROOF OF PROPOSITION 2. Given the update rule from Proposition 1, we have the following form for $p(z|a)$ and $q(z)$,

$$\begin{aligned}
p(z|a) &= \mathcal{B}er\big(z|\sigma(Ca)\big) \\
q(z) &= \mathcal{B}er\big(z|\sigma(C\mathbb{E}\left[a\right])\big).
\end{aligned}$$

The KL divergence inside the expectation then can be rewritten, using the reformulation of the Bernoulli distribution as in the proof of Proposition 1,

$$\begin{aligned}\mathrm{KL}\left(q(z) \parallel p(z|a)\right) &= \mathbb{E}_{q(z)}\left[\log q(z) - \log p(z|a)\right] \\ &= \mathbb{E}_{q(z)}\left[\log\left(\exp(C\mathbb{E}\left[a\right]z)\sigma(-C\mathbb{E}\left[a\right])\right)\right] \\ &\quad - \mathbb{E}_{q(z)}\left[\log\left(\exp(Caz)\sigma(-Ca)\right)\right] \\ &= C\mathbb{E}_{q(z)}\left[z\right]\left(\mathbb{E}\left[a\right] - a\right) + \log\frac{\sigma(-C\mathbb{E}\left[a\right])}{\sigma(-Ca)}.\end{aligned}$$

Taking the expectation $\mathbb{E}_{q(a)}\left[\cdot\right]$ of this expression cancels the first term and we are left with

$$\begin{aligned}\mathbb{E}_{q(a)}\left[\mathrm{KL}\left(q(z) \parallel p(z|a)\right)\right] &= \mathbb{E}_{q(a)}\left[\log\frac{\sigma(-C\mathbb{E}\left[a\right])}{\sigma(-Ca)}\right] \\ &= \mathbb{E}_{q(a)}\left[\log(1+\exp(Ca))\right] - \log(1+\exp(C\mathbb{E}\left[a\right])).\end{aligned}$$

Let $\mathtt{soft}(\cdot) \triangleq \log(1+\exp(\cdot))$ be the softplus function. With this we can rewrite the result as

$$\mathbb{E}_{q(a)}\left[\mathtt{soft}(Ca)\right] - \mathtt{soft}\left(C\mathbb{E}_{q(a)}\left[a\right]\right).$$

In order to compute a closed form approximation to this, note that

$$\mathtt{soft}(x) \approx \begin{cases} x, & \text{for } x \gg 0 \\ 0, & \text{for } x \ll 0 \end{cases} = \max(0, x).$$

That is for a sufficiently large $C$ we have approximately

$$\begin{aligned}&\mathbb{E}_{q(a)}\left[\mathtt{soft}(Ca)\right] - \mathtt{soft}(C\mathbb{E}_{q(a)}\left[a\right]) \\ &\quad\approx C\left(\mathbb{E}_{q(a)}\left[\max(0,a)\right] - \max(0, \mathbb{E}_{q(a)}\left[a\right])\right).\end{aligned}$$

For a normally distributed $a \sim \mathcal{N}(a|\mu,\sigma^2)$, we can calculate the two expectations in this term, giving us the analytical expression

$$C\Big(\mu\Phi\left(\frac{\mu}{\sigma}\right) + \sigma\phi\left(\frac{\mu}{\sigma}\right) - \max(0,\mu)\Big), \tag{3.9}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the pdf and cdf of a standard Normal distribution. For $|\mu| \to \infty$ we have $\phi\left(\frac{\mu}{\sigma}\right) \to 0$ and $\mu\Phi\left(\frac{\mu}{\sigma}\right) \to \max(0,\mu)$, i.e. the overall expression goes to zero as desired. □

This Proposition deserves several comments. The first is that the approximation $\delta_{a>0} \approx \mathcal{B}er\big(a|\sigma(Ca)\big)$ is tight even for decently small $C$ values ($\approx 10$), which allows us to keep a close relationship to the Identity-Heaviside Decomposition and also to the theoretical requirements arising within the proofs, as well as the numerical ones within the implementation. The second comment is that the Proposition relies on the assumption that $p(a)$ follows a normal distribution. In our case, we have for a specific $z_{nj}^l$ that

$$a_{nj}^l = \boldsymbol{w}_{lj}^\top \boldsymbol{h}_n^l = \sum_{i=1}^{n_l} w_{ij}^l h_{ni}^l.$$

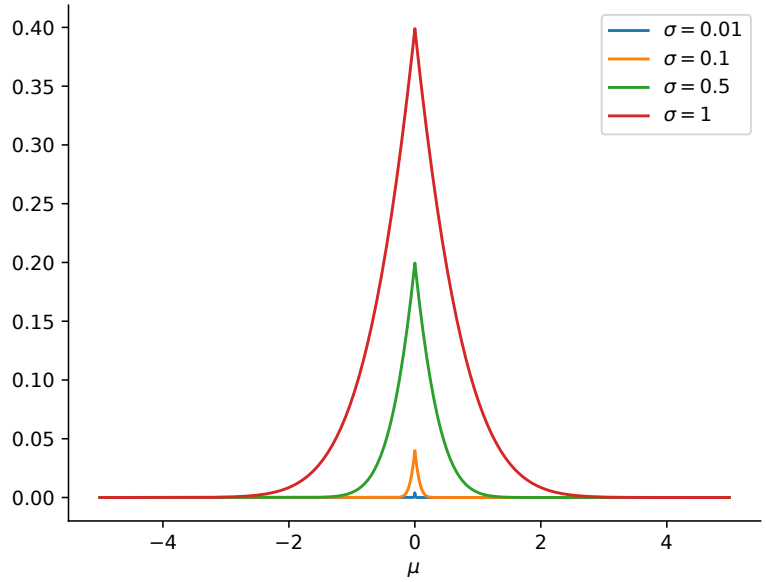



**Figure 3.3: Visualization of Equation** (3.9)**.** The analytical approximation to the expected KL term over the activations, computed with $C = 1$.

This sum allows us to use a central limit theorem argument (Wang and Manning, 2013; Wu et al., 2019), to fulfil this assumption. The relevant $\mu_{nj}^l, \sigma_{nj}^l$ parameters can be computed via a moment-matching approach, analogously to our general derivations above.

As discussed in the proof, the analytical approximation to the term quickly drops to zero as $|\mu| \to \infty$, which we visualize in Figure 3.3. It is tightly concentrated around zero, with $\sigma$ controlling how fast it drops to zero.

How large is the error we make in approximating the softplus to get the final analytical expression? Figure 3.4 shows the corresponding plot if we instead approximate the expectations via samples. The qualitative behaviour is the same, with smaller maxima and somewhat heavier tails.

In the experiments we drop this expected KL term from the ELBO as it becomes negligible for a sufficiently constrained variance. This soft constrained on the variance terms of the layers however is already enforced through the $\text{var}_{q(\mathbf{Z})q_\phi(\mathbf{W})}\left[f(x_n; \mathbf{w})\right]$ term in Equation (3.5) and the pre-activations tend to become either clearly positive or clearly negative.

### 3.1.6 *Convolutions, Pooling, and other Transformations*

*Convolution*

As a linear operation, convolutional layers are directly applicable to the VBP formulation by modifying all the sums between weights and feature maps accordingly with sliding windows. Doing the same will also suffice for the

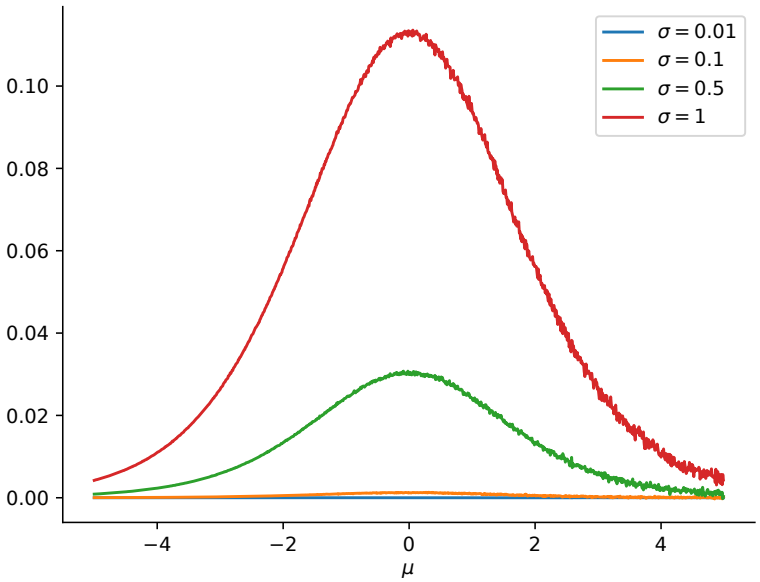



**Figure 3.4: Visualization of** $\mathbb{E}_{q(a)}\left[\mathbf{KL}\left(q(z) \parallel p(z|a)\right)\right]$. The parameters, $\mu$, $\sigma$, and $C$ influence the equation in different ways. The larger $|\mu|$, the closer the expression is to 0. $\sigma$ controls the width and thus how fast the term drops to 0, while $C$ finally scales the whole expression. For this plot, we approximate the softplus by sampling one million points from the corresponding normal distribution for each $\mu, \sigma$ pair with $C = 1$.

calculation of var $[f(\boldsymbol{x}_n; \boldsymbol{w})]$, and the argumentation follows as in the case of fully connected layers.

*Max Pooling*

In the proposed model, one layer affects the next only via sums and products of variables, which is not the case for max-pooling. Even though convolutions are found to be sufficient for building state-of-the-art architectures (Springenberg et al., 2015), where the reduction of resolution can, for example, be accomplished by employing them with a greater stride, we show at the end of this chapter (see Section 3.5.1) with Proposition 3 how VBP can be adapted to allowing for max-pooling, by suitably extending Proposition 1.

*Skip/Residual connections*

As networks grow deeper it was shown that introducing skip connections, where the activation of an earlier layer is concatenated to the input of a later layer, or residual connections, where the activation from a previous layer is added to a later layer, to improve the flow of gradient information becomes very helpful (see, e.g. Ronneberger et al., 2015; He et al., 2016; Huang et al., 2017). We can directly use such connections in the VBP setup with minimal modifications.

*Normalization*

Similarly, with increasing depth, normalization transformations become more relevant. While there is a large variety of methods, the most popular so far still seems to be Batch-Norm (Ioffe and Szegedy, 2015), where a batch of activations is normalized to have mean zero and standard deviation of one. As this introduces a dependency between the normalized activations it cannot be applied directly to the model we propose. Group-Norm (Wu and He, 2018) and other recent propositions suffer from similar problems. However, Actnorm, as suggested by Kingma and Dhariwal (2018) for their Glow model, can be used to allow for normalization in VBP. Here the normalization parameters are initialized by passing a mini-batch of data through the initial net, and adapting the parameters to ensure normalization of each activation. Afterwards, they are no longer updated based on the current data but instead treated as regular hyperparameters. We can update them in the same gradient-based manner as the parameters of the variational distributions.

*Other activation functions*

Lastly, the proposed model can be adapted to any kind of piecewise linear activation function, such as Leaky ReLU (Maas et al., 2013) and its close relation Parametric ReLU (He et al., 2015). Similarly, the hyperbolic tangent activation which is relevant especially in recurrent networks can be approximated piecewise linearly, for example as

$$H(x) = \begin{cases} 1, & \text{if } x > 1 \\ -1, & \text{if } x < -1 \\ x, & \text{else} \end{cases},$$

known as the hard hyperbolic tangent. We show at the end of this chapter in Section 3.5.2 how to extend VBP from the ReLU to such more complex piecewise linear activations.

### 3.1.7 *Classification*

More important than the adaptations to different neural network architecture modifications is the transformation from the regression setting discussed so far to the classification problem. There, we cannot directly decompose the expected log-likelihood

$$\mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[\log p(\boldsymbol{y}|\boldsymbol{W},\boldsymbol{Z},\boldsymbol{X})\right],$$

as we did in Equation (3.5) for regression.

*Binary Classification*

In the case of a binary classification, we can treat $\boldsymbol{y}$ as a vector of latent decision margins and squash it with a binary-output likelihood $p(\boldsymbol{t}|\boldsymbol{y})$. Following Hensman et al. (2013) who used a similar trick for Gaussian processes, the log-marginal likelihood of the resultant bound is given as

$$\begin{aligned}\log p(\boldsymbol{t}|\boldsymbol{X}) &= \log \int p(\boldsymbol{t}|\boldsymbol{y})p(\boldsymbol{y}|\boldsymbol{X})d\boldsymbol{y} \\ &\geq \log \int p(\boldsymbol{t}|\boldsymbol{y})\exp(\mathcal{L}_{\text{elbo}})d\boldsymbol{y} \triangleq \mathcal{L}_{\text{clsf}},\end{aligned}$$

where we used that $\mathcal{L}_{\text{elbo}}$ is a lower bound to the log-marginal likelihood $\log p(\boldsymbol{y}|\boldsymbol{X})$. Choosing a probit likelihood $p(\mathbf{t}|\boldsymbol{y}) = \mathcal{B}er\big(\mathbf{t}|\Phi(\boldsymbol{y})\big)$, where $\Phi$ is the Normal cdf as above, the integral becomes tractable. Referring to the two Kullback-Leibler terms of the $\mathcal{L}_{\text{elbo}}$ term jointly as $\mathcal{C}$ and focusing for notational simplicity on a single data point[7], we have that

$$\begin{aligned}\mathcal{L}_{\text{clsf}} &\triangleq \log \int p(t=1|y)\exp(\mathcal{L}_{\text{elbo}})dy \\ &= \log \int p(t=1|y)\exp\left(\mathbb{E}\left[\log p(y|\boldsymbol{W},\boldsymbol{Z},\boldsymbol{x})\right] - \mathcal{C}\right)dy \\ &= \log \int p(t=1|y)\exp\left(\mathbb{E}\left[\log p(y|\boldsymbol{W},\boldsymbol{Z},\boldsymbol{x})\right]\right)dy - \mathcal{C} \\ &= \log \int \Phi(y)\mathcal{N}\big(y|\mathbb{E}\left[f(\boldsymbol{x};\boldsymbol{w})\right],\beta^{-1}\big)dy + \text{var}\left[f(\boldsymbol{x};\boldsymbol{w})\right] - \mathcal{C} \\ &= \log \sqrt{\beta}\int \Phi(y)\phi\big(\sqrt{\beta}(y-x)\big)dy + \text{var}\left[f(\boldsymbol{x};\boldsymbol{w})\right] - \mathcal{C} \\ &= \log \Phi\left(\frac{\mathbb{E}\left[f(\boldsymbol{x};\boldsymbol{w})\right]}{\sqrt{1+1/\beta}}\right) + \text{var}\left[f(\boldsymbol{x};\boldsymbol{w})\right] - \mathcal{C}.\end{aligned}$$

The expectation and variance expression in the resulting objective can be computed as in the case of regression.

This can then also be trivially extended to multi-class, multi-label classification, by switching to a one-hot encoding for $\boldsymbol{y}_n$ and treating it as $C$ (assuming $C$ classes) one-vs-all problems in the case of multi-class classification. We will use this approach in the next chapter and discuss it in greater detail there.

*Multi-class Classification*

A downside of this one-hot encoded one-vs-all approach is that while it gives

7 As the ELBO decomposes into a sum of independent log-likelihoods, its exponentiated version decomposes into a product of integrals, each of which can be treated independently. This allows for the trivial extension to the case of multiple data points.

us a tractable integral, it ignores the information given by the constraint that in a multi-class setting we know that exactly one class is given. Throughout the experiments we focus on this more common setup with a unique label, i.e. where we have $\boldsymbol{y}_n \in \{0,1\}^C$ with the constraint that $\sum_j y_{nj} = 1$.

This gives us a categorical likelihood

$$\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{Z} \sim \mathcal{C}at\big(\boldsymbol{y}|\zeta(\boldsymbol{f}(\boldsymbol{X};\boldsymbol{w}))\big),$$

where $\zeta(\cdot)$ is the softmax function. In our setup we then have in the ELBO for the data fit term that

$\zeta(\boldsymbol{x})_j \triangleq \frac{\exp(x_j)}{\sum_i \exp(x_i)}$

$$\begin{aligned}
&\mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[\log p(\boldsymbol{y}|\boldsymbol{W},\boldsymbol{Z},\boldsymbol{X})\right] \\
&\quad= \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{Z})q_\phi(\boldsymbol{W})}\left[\boldsymbol{y}_n^\top \boldsymbol{f}(\boldsymbol{x}_n;\boldsymbol{w}) - \texttt{lse}(\boldsymbol{f}(\boldsymbol{x}_n;\boldsymbol{w}))\right] \\
&\quad= \sum_{n=1}^{N} \boldsymbol{y}_n^\top \mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x}_n;\boldsymbol{w})\right] - \mathbb{E}\left[\texttt{lse}(\boldsymbol{f}(\boldsymbol{x}_n;\boldsymbol{w}))\right],
\end{aligned}$$

where $\texttt{lse}(\cdot)$ is the logsumexp function. The first term's expectation is analytically tractable and can be computed as was discussed earlier in Section 3.1.4, while the second expectation is intractable.

$\texttt{lse}(\boldsymbol{x}) \triangleq \log \sum_j \exp(x_j)$

We follow Wu et al. (2019) who propose to use a Taylor approximation to solve this intractability. For a general random variable $\mathbf{a}$ and a function $g(\cdot)$ the second-order Taylor approximation to the expected value is given as

$$\mathbb{E}\left[g(\mathbf{a})\right] \approx g\left(\mathbb{E}\left[\mathbf{a}\right]\right) + \frac{1}{2}\sum_{ij} \operatorname{cov}(a_i, a_j) \frac{\partial^2}{\partial a_i \partial a_j} g\Big|_{\mathbf{a}=\mathbb{E}[\mathbf{a}]}.$$

The first and second order derivatives of $\texttt{lse}(\cdot)$ can be computed as

$$\begin{aligned}
\frac{\partial}{\partial f_i}\texttt{lse}(\boldsymbol{f}) &= \zeta(\boldsymbol{f})_i \\
\frac{\partial^2}{\partial f_i \partial f_j}\texttt{lse}(\boldsymbol{f}) &= \zeta(\boldsymbol{f})_i \delta_{ij} - \zeta(\boldsymbol{f})_i \zeta(\boldsymbol{f})_j,
\end{aligned}$$

where $\delta_{ij} = 1$ if and only if $i = j$.

Combining the two results, and ignoring the covariance terms, we get

$$\mathbb{E}\left[\texttt{lse}(\boldsymbol{f})\right] \approx \texttt{lse}\left(\mathbb{E}\left[\boldsymbol{f}\right]\right) + \frac{1}{2}\sum_{c=1}^{C} \operatorname{var}\left[f_c\right]\Big(\zeta\big(\mathbb{E}\left[\boldsymbol{f}\right]\big)_c - \zeta\big(\mathbb{E}\left[\boldsymbol{f}\right]\big)_c^2\Big),$$

with $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}_n;\boldsymbol{w})$. In this final expression the expectation and variance terms can in turn be computed as in the regression case.

*Posterior predictive for classification*

Finally, for the posterior predictive we can follow the same argument this time approximating the softmax function instead of the logsumexp with a Taylor approximation. However, we have observed that using samples in the last step of computing the softmax is cheap enough after computing all earlier

layers in the sampling-free approach. As it removes the need for yet another approximation, we will therefore rely on samples in the prediction of the classification experiments. The posterior predictive can then be approximated efficiently for some test observation $\boldsymbol{x}_*$ as

$$\begin{aligned} p(\boldsymbol{y}_*|\boldsymbol{x}_*, \mathcal{D}) &= \iint p(\boldsymbol{y}_*|\boldsymbol{x}_*, \boldsymbol{W}, \boldsymbol{Z}) p(\boldsymbol{Z}, \boldsymbol{W}|\mathcal{D}) \, \mathrm{d}\boldsymbol{Z} \, \mathrm{d}\boldsymbol{W} \\ &\approx \int \mathcal{C}at(\boldsymbol{y}_*|\zeta(\boldsymbol{f}_*^L)) \mathcal{N}\left(\boldsymbol{f}_*^L | \mathbb{E}\left[\boldsymbol{f}_*^L\right], \mathrm{var}\left[\boldsymbol{f}_*^L\right]\right) \mathrm{d}\boldsymbol{f}_*^L \\ &\approx \frac{1}{S} \sum_{s=1}^{S} \mathcal{C}at(\boldsymbol{y}_*|\zeta(\boldsymbol{f}_*^{L(s)})), \end{aligned}$$

where $\boldsymbol{f}_*^L = f(\boldsymbol{x}_*; \boldsymbol{w})$, and we approximate the final integral by taking $S$ samples $\boldsymbol{f}_*^{L(s)} \sim \mathcal{N}\left(\boldsymbol{f}_*^{L(s)} | \mathbb{E}\left[\boldsymbol{f}_*^L\right], \mathrm{var}\left[\boldsymbol{f}_*^L\right]\right)$.

## 3.2 RELATED WORK

### 3.2.1 *Relation to CLT based Approaches*

VBP is most closely related to two other state-of-the-art sampling-free approaches. These are *Deterministic Variational Inference* (DVI) (Wu et al., 2019)[8] and *Probabilistic Backpropagation* (PBP) (Hernández-Lobato and Adams, 2015). PBP follows an assumed density filtering approach instead of variational inference but relies as DVI does on a central limit theorem (CLT) argument as a major part of the pipeline.

We will also use such a CLT argument in the following two chapters and discuss the approach in greater detail there. However, to give an intuition as necessary to evaluate the subsequent experiments, the general idea is the following. Based on the observation that the pre-activations of each layer

$$f_{nj}^l = \boldsymbol{w}_{lj}^\top \boldsymbol{h}_n^l = \sum_{i=1}^{n_l} w_{ij}^l h_{ni}^l$$

are approximately normally distributed, we can perform a moment matching approach to compute its mean and variance parameters from the earlier layers similar to what we rely on for VBP. Given this normal approximation, the mean and variance of the post-activation feature maps $h = \max(0, f)$ are then tractable for ReLU activations (Frey and Hinton, 1999),

$$\begin{aligned} \mathbb{E}_{\mathcal{N}(f|\mu,\sigma^2)}\left[\max(0, f)\right] &= \mu\Phi\left(\frac{\mu}{\sigma}\right) + \sigma\phi\left(\frac{\mu}{\sigma}\right), \\ \mathrm{var}\left[\max(0, f)\right] &= (\mu^2 + \sigma^2)\Phi\left(\frac{\mu}{\sigma}\right) + \mu\sigma\phi\left(\frac{\mu}{\sigma}\right) - \mathbb{E}\left[h\right]^2, \end{aligned}$$

and can then be propagated forward to the next layer. $\Phi(\cdot)$ and $\phi(\cdot)$ here refer to the cdf and pdf of a standard normal distribution.

8 For simplicity we focus only on the homoscedastic, mean-field variation of DVI here.

Due to our decomposition approach we do not rely on the CLT. If we assume a similar moment matching for the pre-activations, we would get the corresponding expectation and variances as

$$\begin{aligned}\mathbb{E}[h] &= \mathbb{E}[z]\,\mathbb{E}[f] = \mathbb{E}[z]\,\mu,\\ \operatorname{var}[h] &= \mathbb{E}\left[z^2\right]\operatorname{var}[f] + \operatorname{var}[z]\,\mathbb{E}[f]^2\\ &= \mathbb{E}\left[z^2\right]\sigma^2 + \operatorname{var}[z]\,\mu^2 \approx \mathbb{E}[z]\,\sigma^2,\end{aligned}$$

where the value of $\mathbb{E}[z] = \sigma(C\mu)$ decides whether the mean and variance of the pre-activation $f$ are propagated or blocked. Comparing the two, we can see that large $|\mu|$ values then give similar mean, variances to be propagated to the next layer, while for smaller values the behaviour differs.

### 3.2.2 *Other related work*

Several approaches have been introduced in the recent decades for approximating the intractable posterior of BNNs. One line of research is model-based Markov Chain Monte Carlo, represented by methods such as Hamiltonian Monte Carlo (HMC) (Neal, 2010) and Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011). Chen et al. (2014) adapted HMC to stochastic gradients by quantifying the entropy overhead stemming from the stochasticity of mini-batch selection.

While being actively used for a broad spectrum of probabilistic models, the successful application of variational inference to deep neural nets has only occurred recently. The earliest study to infer a BNN with variational inference (Hinton and Camp, 1993) was applicable for only one hidden layer. This limitation has been overcome by Graves (2011) approximating the intractable expectations by numerical integration. Further scalability has been achieved after stochastic gradient variational Bayes (SGVB) has been made applicable to BNN inference using weight reparameterizations (Kingma and Welling, 2014; Rezende et al., 2014).

Dropout (Srivastava et al., 2016) has strong connections to the variational inference of BNNs. Gal and Ghahramani (2016b) developed a theoretical link between a dropout network and a deep Gaussian process (Damianou and Lawrence, 2013) inferred by variational inference. Kingma et al. (2015) showed that extending the Bayesian model selection interpretation of Gaussian Dropout with a log-uniform prior on weights leads to a BNN inferred by SGVB. Our proposed model can also be interpreted as an input-dependent dropout (Ba and Frey, 2013; Lee et al., 2018b) applied to a linear net. Yet it differs from them and the standard dropout. The masking variable always shuts down negative activations, hence does not serve as a regularizer but instead implements the ReLU nonlinearity.

A fundamental step in reducing ELBO gradient variance has been taken by Kingma et al. (2015) with local reparameterization, which suggests taking the Monte Carlo integrals by sampling the linear activations instead of the

weights. Further variance reduction has been achieved by defining the variances of the variational distribution factors as free parameters and the dropout rate as a function of them (Molchanov et al., 2017). Theoretical treatments of the same problem have also been recently studied (see e.g. Miller et al., 2017; Roeder et al., 2017).

SGVB has been introduced initially for fully factorized variational distributions, which limites the feasible posteriors that one can infer. Strategies for improving the approximation quality of variational BNN inference include employment of structured versions of dropout matrix normals (Louizos and Welling, 2016), repetitive invertible transformations of latent variables, known as normalizing flows, (Rezende and Mohamed, 2015) and their application to variational dropout (Louizos and Welling, 2017). Wu et al. (2019) use the CLT argument to move beyond mean-field variational inference and demonstrate how to adapt it to learn layerwise covariance structures for the variational posteriors.

Lastly, there is active research on enriching variational inference using its interpolative connection to expectation propagation (Hernández-Lobato and Adams, 2015; Li and Turner, 2016; Li and Gal, 2017).

## 3.3 EXPERIMENTS

We evaluate the proposed model on a variety of regression and classification data sets. Details on hyperparameters and architectures not provided in the main text can be found in the appendix.[9]

### 3.3.1 *Regression*

For the regression experiments, we follow the experimental setup introduced by Hernández-Lobato and Adams (2015) which consists of evaluating the model performance on nine UCI benchmark data sets with a normal likelihood $\mathcal{N}(\boldsymbol{y}|f(\boldsymbol{X};\boldsymbol{w}), \beta^{-1}\mathbb{1})$. We train a BNN with one hidden layer of 50 units for each data set, except for the larger `protein` data set which uses 100 hidden neurons. Each data set is twenty times randomly split into train and test data, consisting of 90% and 10% of the data, respectively. We optimize the model using the Adam optimizer (Kingma and Ba, 2015) with their proposed default parameters and a learning rate of $\lambda = 0.01$.

We compare against the two sampling-free approaches introduced above, *Probabilistic Back-Propagation* (PBP) (Hernández-Lobato and Adams, 2015) and *Deterministic Variational Inference* (DVI) (Wu et al., 2019), as well as the sampling-based *Variational Dropout* (VarOut) (Kingma et al., 2015) in the formulation of Molchanov et al. (2017). The results for these baselines are cited from the respective papers. We rely on our own implementation for VarOut, replacing the improper log-uniform prior with a proper Gaussian prior to avoid a possible improper posterior (Hron et al., 2017).

9 A reference implementation is provided at `https://github.com/manuelhaussmann/vbp`.

**Table 3.1: Regression Results**. Reported are the average test log-likelihood ± standard error over 20 random train/test splits. The best performing method is marked bold. The $N/d$ column gives the number of data points and the input feature dimension for each data set.

| data set | N/d | DVI | PBP | VarOut | VBP |
|---|---|---|---|---|---|
| boston | 506/13 | $-2.58_{\pm 0.04}$ | $\mathbf{-2.57_{\pm 0.09}}$ | $-2.59_{\pm 0.03}$ | $-2.59_{\pm 0.03}$ |
| concrete | 1030/8 | $-3.23_{\pm 0.01}$ | $-3.16_{\pm 0.02}$ | $-3.18_{\pm 0.02}$ | $\mathbf{-3.15_{\pm 0.02}}$ |
| energy | 768/8 | $-2.09_{\pm 0.06}$ | $-2.04_{\pm 0.02}$ | $-1.25_{\pm 0.05}$ | $\mathbf{-1.11_{\pm 0.07}}$ |
| kin8nm | 8192/8 | $1.01_{\pm 0.01}$ | $0.90_{\pm 0.01}$ | $1.02_{\pm 0.01}$ | $\mathbf{1.04_{\pm 0.01}}$ |
| naval | 11934/16 | $\mathbf{5.84_{\pm 0.06}}$ | $3.73_{\pm 0.01}$ | $5.52_{\pm 0.04}$ | $5.79_{\pm 0.07}$ |
| power | 9568/4 | $\mathbf{-2.82_{\pm 0.00}}$ | $-2.84_{\pm 0.01}$ | $-2.83_{\pm 0.01}$ | $-2.85_{\pm 0.01}$ |
| protein | 45730/9 | $-2.94_{\pm 0.00}$ | $-2.97_{\pm 0.00}$ | $\mathbf{-2.92_{\pm 0.00}}$ | $\mathbf{-2.92_{\pm 0.00}}$ |
| wine | 1599/11 | $\mathbf{-0.96_{\pm 0.01}}$ | $-0.97_{\pm 0.01}$ | $\mathbf{-0.96_{\pm 0.01}}$ | $\mathbf{-0.96_{\pm 0.01}}$ |
| yacht | 308/6 | $\mathbf{-1.41_{\pm 0.03}}$ | $-1.63_{\pm 0.02}$ | $-1.65_{\pm 0.05}$ | $-1.54_{\pm 0.06}$ |

To learn the observation precision $\beta$ for our model and the VarOut baseline, we follow a type-II Maximum Likelihood approach and after each training epoch update it so that the ELBO is maximized, which reduces to choosing $\beta$ to maximize the data fit, i.e.

$$\beta^* = \arg\max_{\beta} \mathbb{E}_{q(\theta)} \left[ \log p(\boldsymbol{y}|\theta, \boldsymbol{x}) \right],$$

which for $p(y_n|\theta, \boldsymbol{x}_n) = \mathcal{N}(y_n|f(\boldsymbol{x}_n;\theta), \beta^{-1})$ takes the form

$$\frac{1}{\beta^*} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{q(\theta)} \left[ \left( y_n - f(\boldsymbol{x}_n;\theta) \right)^2 \right],$$

This expression is either evaluated via samples for VarOut or deterministically for VBP as we have shown above. One could also introduce a hyperprior over the prior weight precisions, and learn them via a type-II approach as done in DVI (Wu et al., 2019). Instead we use a fixed normal prior $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1}\mathbb{1})$. We set $\alpha = 10$ and use $\beta = 1$ as the initial observation precision, observing quick convergence in general.

We summarize the average test log-likelihood together with the standard error over twenty random splits in Table 3.1. VBP either outperforms the baselines or performs competitively with them.

### 3.3.2 *Classification*

Our main classification experiment is an evaluation on four standard image classification data sets of increasing complexity: MNIST (LeCun et al., 1998), FashionMNIST (Xiao et al., 2017), CIFAR-10, and CIFAR-100 (Krizhevsky and Hinton, 2009). The goal is to evaluate how the three variational inference based approaches of either taking samples (VarOut), relying on CLT (DVI), or the ReLU decomposition (VBP) compare as the depth of the BNN increases from the small regression net.

**Table 3.2: Classification Results.** Average classification error rate and test log-likelihoods ± standard deviation over five runs

| | Average Error (in %) | | | Average Test Log-Likelihood (in %) | | |
|---|---|---|---|---|---|---|
| | VarOut | DVI | VBP | VarOut | DVI | VBP |
| Mnist | 1.12±0.05 | 0.97±0.06 | **0.85**±0.06 | **−0.03**±0.00 | **−0.03**±0.00 | **−0.03**±0.00 |
| FashionMNIST | 10.81±0.24 | 10.33±0.07 | **10.10**±0.16 | −0.30±0.00 | −0.29±0.00 | **−0.28**±0.00 |
| Cifar-10 | 33.40±1.06 | 35.15±1.13 | **31.33**±1.01 | −0.95±0.03 | −1.00±0.03 | **−0.90**±0.03 |
| Cifar-100 | 62.85±1.43 | 66.21±1.14 | **60.97**±1.61 | −2.44±0.06 | −2.61±0.06 | **−2.37**±0.08 |

The architecture we use is a LeNet5 sized net consisting of two convolutional layers and two fully connected layers, with more filters/units per layer for the two CIFAR data sets. As discussed in Section 3.1.6, VBP can handle max-pooling layers, but they require careful tracking of indices between the data fit and variance terms, which comes at some extra run time cost in present deep learning libraries. As they are also intractable for DVI, we do not use pooling in the architecture. Instead, we provide a reference implementation on how to do this but stick in the experiments with strided convolutions following the recent trend of "all-convolutional-nets" (Springenberg et al., 2015; Yu et al., 2017; Redmon and Farhadi, 2018).

We compare VBP against VarOut and our implementation of DVI as Wu et al. (2019) stick in their publication to the regression setup discussed above. To improve its scalability and keep it comparable to VBP and VarOut, we use the mean-field setup of DVI. In order to ensure maximal comparability between the three methods all of them share the same normal prior $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1}\mathbb{1})$ ($\alpha = 100$), initialization and other hyperparameters. They are optimized with the Adam optimizer (Kingma and Ba, 2015) and a learning rate of $\lambda = 0.001$ over 100 epochs.

We summarize the results in Table 3.2 reporting the average predictive error in per cent and the average test log-likelihood over five runs. We observe that DVI, making efficient use of the CLT based moment-matching approach, improves upon the sampling-based VarOut on the two easier data sets, while it struggles on the two more complex CIFAR task. VBP can deal with the increasing width from (Fashion)MNIST to CIFAR-{10,100} a lot better, improving upon both VarOut and DVI on all four data sets. As the depth increases from the regression to the classification experiment the performance difference that was small for the shallow network becomes more and more pronounced.

### 3.3.3 *Online Learning Comparison*

As the final experiment, we follow the setup of Kandemir (2018), i.e. the gradient-free method with closed-form updates. They argue that their focus on closed-form updates instead of relying on gradients gives them an advantage in an online learning setup that has the constraint of allowing only a single pass over the data. They report results on MNIST and CIFAR-10, using a net with a single hidden layer of 500 units for MNIST and a two hidden layer net with

**Table 3.3: Online Learning Results**. Average test set accuracy (in %) ± standard deviation over ten runs.

| | MNIST | CIFAR-10 |
|---|---|---|
| V-ReLU-Net | 92.8±0.2 | 47.1±0.2 |
| VarOut | **96.7**±**0.2** | 47.8±0.3 |
| DVI | 96.6±0.2 | **48.7**±**0.4** |
| VBP | 96.6±0.2 | 48.3±0.4 |

2048/1024 units for CIFAR-10. We run the three variational inference BNNs in the same setup. While the closed-form updates of V-ReLU-Net (Kandemir, 2018) have the advantage of removing the need for gradients, the required mean-field approximation over the layers substantially constrains it compared to the more flexible VBP structure. We summarize the average test set accuracy in per cent over ten runs in Table 3.3

## 3.4 CONCLUSION

Our experiments demonstrate that the Identity-Heaviside decomposition and the variance back-propagation we propose offer a powerful alternative to other recent deterministic approaches of training deterministic BNNs.

Following the No-Free-Lunch theorem, our closed-form available ELBO comes at the expense of several restrictions, such as a fully factorized approximate posterior, sticking to ReLU activations, and inapplicability of Batch Normalization (Ioffe and Szegedy, 2015). An immediate implication of this work is to explore ways to relax the mean-field assumption and incorporate normalizing flows without sacrificing from the closed-form solution. Because Equations (3.6) and (3.7) extend easily to dependent variables after adding the covariance of each variable pair as done by Wu et al., 2019, our formulation is adaptable to structured variational inference schemes without major theoretical obstacles. Further extensions that are directly applicable to our construction are, as discussed, the inclusion of residual (He et al., 2016) and skip connections (Huang et al., 2017), which is an interesting direction for future work as it will allow this approach to scale to deeper architectures.

## 3.5 FURTHER DETAILS

This final section of the chapter discusses the necessary adaption of VBP to allow for max-pooling and more activation functions. We then discuss the restriction we relied on throughout the chapter of computing only the variance terms, while dropping any covariance structure on a layer's post-activations. The section finishes with some further experimental details to ensure reproducibility.

### 3.5.1 *Max Pooling within VBP*

The following proposition discusses how to incorporate max-pooling layers within the model. It is an extension of Proposition 1 from the main text.

PROPOSITION 3. *For a generative model including the generative process excerpt as below*

$$\begin{aligned} &\vdots \\ a &\sim p(a), \\ b &\sim p(b), \\ c|a,b &= \max(a,b), \\ d|c &\sim p(b|h(c)), \\ &\vdots \end{aligned}$$

*with some arbitrary function $h(c)$, when mean-field variational Bayes is performed with an approximate distribution $Q = \cdots q(a)q(b)q(c)\cdots$, the following identities hold*

(i) $\mathbb{E}[c] = \mathbb{E}[\max(a,b)] = \max(\mathbb{E}[a], \mathbb{E}[b])$.

(ii) $\operatorname{var}[c] = \begin{cases} \operatorname{var}[a], & \text{if } \mathbb{E}[a] \geq \mathbb{E}[b] \\ \operatorname{var}[b], & \text{else} \end{cases}$

PROOF. We first rewrite the generative process as

$$\begin{aligned} &\vdots \\ a &\sim p(a), \\ b &\sim p(b), \\ z &\sim \delta_{a-b>0}, \\ c|a,b,z &= a\,z + b\,(1-z), \\ d|c &\sim p(b|h(c), \\ &\vdots \end{aligned}$$

For a variational distribution $Q = \cdots q(a)q(b)q(z)q(c)\cdots$ the optimal update for $z$ is then given as

$$q(z) \leftarrow \delta_{\mathbb{E}[a]-\mathbb{E}[b]>0},$$

which we get from applying Proposition 1. Then

$$\begin{aligned} \mathbb{E}[c] &= \mathbb{E}[\max(a,b)] \\ &= \mathbb{E}[a]\,\mathbb{E}[z] + \mathbb{E}[b]\,\mathbb{E}[1-\mathbb{E}[z]] \\ &= \max(\mathbb{E}[a], \mathbb{E}[b]), \end{aligned}$$

which satisfies (i).

In order to get (ii), we decompose the variance term

$$\begin{aligned}\operatorname{var}[c] &= \operatorname{var}[az] + \operatorname{var}[b(1-z)] \\ &= \mathbb{E}[z]\operatorname{var}[a] + (1-\mathbb{E}[z])\operatorname{var}[b]\,,\end{aligned}$$

using that $z$ has zero variance. As $z = 1$ for $\mathbb{E}[a] > \mathbb{E}[b]$, we get the desired result for the variance. □

This outcome can, in turn, be directly extended to $\max(\cdot)$ functions with an arbitrary number of inputs by using the following recursive identity

$$\max(a_1, \ldots, a_n) = \max(\max(a_1, \ldots, a_{n-1}), a_n).$$

Consequently, we can plug a max-pooling layer into our framework. The data-fit term in the ELBO will use (i) while the variance term, on the other hand, will rely on (ii).

### 3.5.2 *Extension to other Activation Functions*

While we introduce and discuss the Identity-Heaviside decomposition in the main text solely for the popular ReLU activation function, the approach can be extended straightforwardly for other piecewise linear activation functions.

*PReLU/Leaky ReLU*

One extension to ReLUs is to allow for a non-zero gradient not only for positive input but also if $x < 0$. Leaky ReLU (Maas et al., 2013) and Parametric ReLU (PReLU) (He et al., 2015) have the following structure

$$g(x) \triangleq \begin{cases} x & \text{if } x > 0 \\ ax & \text{if } x \leq 0 \end{cases},$$

where $a$ is either fixed to a small value as in the Leaky ReLU or a learnable parameter as in PReLU. Following the notation from the main text, with $\boldsymbol{z}^l$ as before for the $l$-th layer, we define

$$\boldsymbol{c}^l = a + (1-a) \cdot \boldsymbol{z}^l.$$

This gives us the desired structure for the post-activation feature vector

$$\boldsymbol{h}^l = \boldsymbol{f}^l \circ \boldsymbol{c}^l.$$

The mean and variance of $c_i^l$ remain tractable with

$$\begin{aligned}\mathbb{E}\left[c_i^l\right] &= a + (1-a)\mathbb{E}\left[z_i^l\right], \\ \operatorname{var}\left[c_i^l\right] &= (1-a)^2 \operatorname{var}\left[z_i^l\right].\end{aligned}$$

The rest of the equations can be analogously updated and remain tractable. Depending on whether $a$ is supposed to be fixed or adapted, it can then be updated via gradient descent just as the other parameters.

*Tanh Activation*

In order to extend the approach to sigmoid activation functions, like $\tanh(\cdot)$ or $\sigma(x) = 1/(1+\exp(-x))$, one can similarly extend the approach by instead of considering a two piece linear split $(-\infty, 0)$ and $[0, \infty)$ splitting intothree linear pieces $(-\infty, -\varepsilon)$, $[-\varepsilon, \varepsilon)$, and $[\varepsilon, \infty)$, for some chosen $\varepsilon$ instead, by extending the current $\boldsymbol{z}$ definition suitably.

Consider for example the hard hyperbolic tangent as an approximation to $\tanh(\cdot)$. It is defined as

$$H(x) \triangleq \begin{cases} 1, & \text{if } x > 1 \\ -1, & \text{if } x < -1 \\ x, & \text{else} \end{cases}.$$

The post-activation of the $l$-th layer is then, dropping the layer reference from the notation, given as

$$\boldsymbol{h} = \boldsymbol{u} - \boldsymbol{v} + \boldsymbol{f}(1-\boldsymbol{u})(1-\boldsymbol{v}),$$

where we defined $\boldsymbol{u} = \mathbb{1}(\boldsymbol{f} - 1)$ and $\boldsymbol{v} = \mathbb{1}(-\boldsymbol{f} - 1)$. The Bernoulli approximation for the two can then be given in analogy to the one for $z$ in the main part of the chapter via

$$u|f \sim \mathcal{B}er\big(u|\sigma(C(f-1))\big), \qquad v|f \sim \mathcal{B}er\big(u|\sigma(-C(f+1))\big).$$

Following the derivations in the main text we have the expectation as

$$\mathbb{E}[\boldsymbol{h}] = \mathbb{E}[\boldsymbol{u}] + \mathbb{E}[\boldsymbol{v}] + \mathbb{E}[\boldsymbol{f}]\,(1 - \mathbb{E}[\boldsymbol{u}])(1 - \mathbb{E}[\boldsymbol{v}]),$$

where each term is directly tractable. The variance in turn is given as

$$\begin{aligned} \operatorname{var}[\boldsymbol{h}] &= \operatorname{var}[u + v + f(1-u)(1-v)] \\ &= \operatorname{var}[u] + \operatorname{var}[v] + \operatorname{var}[f(1-u)(1-v)] \\ &\quad + 2\operatorname{cov}(u, f(1-u)(1-v)) + 2\operatorname{cov}(v, f(1-u)(1-v)). \end{aligned}$$

Again, every term is tractable, however at an increased computational cost. We have that the third variance term is given via

$$\begin{aligned} &\operatorname{var}[f(1-u)(1-v)] = \mathbb{E}\left[(f(1-u)(1-v))^2\right] - (\mathbb{E}[f(1-u)(1-v)])^2 \\ &= \mathbb{E}\left[f^2\right]\mathbb{E}\left[(1-u)^2\right]\mathbb{E}\left[(1-v)^2\right] - \mathbb{E}[f]^2\,\mathbb{E}[(1-u)]^2\,\mathbb{E}[(1-v)]^2 \\ &= \mathbb{E}\left[f^2\right](1 - \mathbb{E}[u])(1 - \mathbb{E}[v]) - \mathbb{E}[f]^2\,(1 - \mathbb{E}[u])^2(1 - \mathbb{E}[v])^2. \end{aligned}$$

The two covariance terms can similarly be decomposed as

$$\begin{aligned} &\operatorname{cov}(u, f(1-u)(1-v)) = \mathbb{E}[uf(1-u)(1-v)] - \mathbb{E}[u]\,\mathbb{E}[f(1-u)(1-v)] \\ &= \mathbb{E}[u(1-u)]\,\mathbb{E}[f]\,\mathbb{E}[(1-v)] - \mathbb{E}[u]\,\mathbb{E}[f]\,\mathbb{E}[(1-u)]\,\mathbb{E}[(1-v)] \\ &= -\mathbb{E}[u]\,\mathbb{E}[f]\,(1 - \mathbb{E}[u])(1 - \mathbb{E}[v]), \end{aligned}$$

where we use that $\mathbb{E}[u(1-u)] = 0$. The second covariance term is analogously given as

$$\operatorname{cov}(u, f(1-u)(1-v)) = -\mathbb{E}[v]\,\mathbb{E}[f]\,(1 - \mathbb{E}[v])(1 - \mathbb{E}[u]).$$

The variational posterior distributions for $q(u)$ and $q(v)$ are still analytically tractable, which follows directly by a minor adaptation of Proposition 1.

### 3.5.3 *Covariance Terms*

As we discussed in the main text when deriving the variance terms, we need to restrict ourselves in practice to only computing and propagating the variance of each layer's pre/post activations, limiting us essentially to a diagonal covariance matrix. This section gives the necessary derivations for a working with complete covariance matrices instead.

Focusing solely on the $l$-th layer and dropping the $l$ from the notation for simplicity, we have that $\mathbb{E}[\boldsymbol{h}]$ is independent of the restriction and can be computed as before. For the covariance between $h_i, h_j$ however, we have for $i \neq j$ that it can be computed as

$$\begin{aligned}
\operatorname{cov}(h_i, h_j) &= \mathbb{E}[h_i h_j] - \mathbb{E}[h_i]\,\mathbb{E}[h_j] \\
&= \mathbb{E}[z_i f_i z_j f_j] - \mathbb{E}[z_i f_i]\,\mathbb{E}[z_j f_j] \\
&= \mathbb{E}[z_i]\,\mathbb{E}[z_j]\,\mathbb{E}[f_i f_j] - \mathbb{E}[z_i]\,\mathbb{E}[z_j]\,\mathbb{E}[f_i]\,\mathbb{E}[f_j] \\
&= \mathbb{E}[z_i]\,\mathbb{E}[z_j]\operatorname{cov}(f_i, f_j)\,,
\end{aligned}$$

where we used the variational approximation of independence between $z_i$ and $z_j$. The two expectations are trivially given, while the remaining covariance term can be computed as follows. Using the slight notational overloading of the new $\boldsymbol{h}$ variables to refer to the layer before and switching to a vectorial formulation, we have

$$\begin{aligned}
\operatorname{cov}(f_i, f_j) &= \operatorname{cov}\left(\boldsymbol{w}_i^\top \boldsymbol{h}, \boldsymbol{w}_j^\top \boldsymbol{h}\right) \\
&= \mathbb{E}\left[(\boldsymbol{w}_i^\top \boldsymbol{h})(\boldsymbol{w}_j^\top \boldsymbol{h})\right] - \mathbb{E}\left[\boldsymbol{w}_i^\top \boldsymbol{h}\right]\mathbb{E}\left[\boldsymbol{w}_j^\top \boldsymbol{h}\right] \\
&= \mathbb{E}\left[\boldsymbol{w}_i^\top\right]\mathbb{E}\left[\boldsymbol{h}\boldsymbol{h}^\top\right]\mathbb{E}\left[\boldsymbol{w}_j\right] - \mathbb{E}[\boldsymbol{w}_i]^\top\,\mathbb{E}[\boldsymbol{h}]\,\mathbb{E}[\boldsymbol{h}]^\top\,\mathbb{E}\left[\boldsymbol{w}_j\right] \\
&= \mathbb{E}[\boldsymbol{w}_i]^\top \operatorname{cov}(\boldsymbol{h}, \boldsymbol{h})\,\mathbb{E}\left[\boldsymbol{w}_j\right]\,,
\end{aligned}$$

where we used the variational approximation of independence between the weights $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$. The variance for $i = j$ can be computed as in the main text in this vectorial form as

$$\begin{aligned}
\operatorname{var}[f_i] &= \mathbb{E}\left[(\boldsymbol{w}_i^\top \boldsymbol{h})(\boldsymbol{w}_i^\top \boldsymbol{h})\right] - \mathbb{E}\left[\boldsymbol{w}_i^\top \boldsymbol{h}\right]\mathbb{E}\left[\boldsymbol{w}_i^\top \boldsymbol{h}\right] \\
&= \operatorname{tr}\left(\mathbb{E}\left[\boldsymbol{w}_i \boldsymbol{w}_i^\top\right]\mathbb{E}\left[\boldsymbol{h}\boldsymbol{h}^\top\right]\right) - \mathbb{E}\left[\boldsymbol{w}_i^\top \boldsymbol{h}\right]\mathbb{E}\left[\boldsymbol{w}_i^\top \boldsymbol{h}\right] \\
&= \operatorname{tr}\left(\mathbb{E}\left[\boldsymbol{w}_i \boldsymbol{w}_i^\top\right]\left(\mathbb{E}[\boldsymbol{h}]\,\mathbb{E}[\boldsymbol{h}]^\top + \operatorname{var}[\boldsymbol{h}]\right)\right) \\
&\quad - \mathbb{E}[\boldsymbol{w}_i]^\top\,\mathbb{E}[\boldsymbol{h}]\,\mathbb{E}[\boldsymbol{w}_i]^\top\,\mathbb{E}[\boldsymbol{h}] \\
&= \operatorname{tr}\Big(\left(\mathbb{E}\left[\boldsymbol{w}_i \boldsymbol{w}_i^\top\right] - \mathbb{E}[\boldsymbol{w}_i]\,\mathbb{E}[\boldsymbol{w}_i]^\top\right)\mathbb{E}[\boldsymbol{h}]\,\mathbb{E}[\boldsymbol{h}]^\top \\
&\quad + \mathbb{E}\left[\boldsymbol{w}_i \boldsymbol{w}_i^\top\right]\operatorname{var}[\boldsymbol{h}]\Big) \\
&= \operatorname{tr}\left(\operatorname{var}[\boldsymbol{w}_i]\,\mathbb{E}[\boldsymbol{h}]\,\mathbb{E}[\boldsymbol{h}]^\top + \mathbb{E}\left[\boldsymbol{w}_i \boldsymbol{w}_i^\top\right]\operatorname{var}[\boldsymbol{h}]\right)\,,
\end{aligned}$$

where we use the trick of first applying the trace operator $\operatorname{tr}(\cdot)$ to a scalar and then using the linearity of the expectation. Reverting this trace gives us the result from the main discussion.

**Table 3.4:** The Strided LeNet architecture. This table gives the architecture used in the classification experiments of the main text. For Cifar-10/Cifar-100 the convolutional layers get instead 192 filters each and the number of neurons in the penultimate fully connected layer is increased to 1000. The activations between all layers are ReLUs.

| STRIDED LENET5 |
|---|
| Convolution $(5 \times 5)$ with 20 channels, stride 2 |
| Convolution $(5 \times 5)$ with 50 channels, stride 2 |
| Linear with 500 neurons |
| Linear with $n_{\text{class}}$ neurons |

These derivations show that computing and propagating full covariance matrices is tractable mathematically. They also demonstrate that while one could consider switching from a pure mean-field factorization over the weights and still keep tractable computations, e.g. by considering layerwise independence and within a layer between different $\boldsymbol{w}_i, \boldsymbol{w}_j$, the computational cost both in terms of memory and runtime make this an infeasible approach in practice.

### 3.5.4 *Further Experimental Details*

REGRESSION SETUP. For the regression experiments we follow the common experimental setup for the evaluation of BNN regression as introduced by Hernández-Lobato and Adams (2015). The neural net consists of one hidden layer of 50 for all data sets except for `protein` where it is doubled to 100 neurons. For each data set we use the Adam optimizer with a learning rate of $\lambda = 0.01$, with varying batch sizes depending on the data set size to ensure a roughly equal number of gradient steps. `boston`, `energy` and `yacht` get a batch size of 16. `concrete` and `wine` use 32 instances per batch, `power`, `kin8nm`, and `naval` use 64 and `protein` finally gets 256 instances.

CLASSIFICATION SETUP. The convolutional neural architecture we use in the convolutional classification experiments is a modified version of the classical LeNet5, without pooling layers but strided convolutions instead. The version used with MNIST and FashionMNIST is summarized in Table 3.4.

For Cifar-10/100 we increase the width of the convolutional layers to 192 channels each and the linear one to 1000, following Gal and Ghahramani (2016a). All hidden layers are followed by ReLU activations. As the optimizer we use again Adam with the default hyperparameters and a learning rate of $\lambda = 0.001$ for each data set.

ONLINE CLASSIFICATION. We follow the architecture setup prescribed by Kandemir (2018). That is, one hidden layer of 500 units for the MNIST experiment and a two hidden layer network with 2048 units in the first and 1024 units in the second layer for the Cifar-10 experiment.

INITIALIZATION OF THE VARIATIONAL PARAMETERS. The initialization scheme is shared between all architectures, experiments and the three BNN variants: VBP, VarOut, and DVI. The variational means of the weights follow the common initialization of He et al. (2015). The logarithms of their variances are sampled from $\mathcal{N}(-9, 1e-3)$.

# 4

# DEEP ACTIVE LEARNING WITH ADAPTIVE ACQUISITION

Deep Learning offers great potential. However, it suffers also from great hunger for data. Active Learning (Settles, 2012) tackles whether this can be optimized, i.e. is every data point created equal or are different data points of varying usefulness for the model to learn.

The choice of a proper acquisition function, i.e. how to select this "usefulness" is a model design choice similar to other selection choices, such as the architecture, learning rate or other hyperparameters. One can summarize these choices under the keyword of model selection. Model selection is treated as a standard performance-boosting step in many machine learning applications. Once all other properties of a learning problem are fixed, the model is selected by grid search on a held-out validation set. However, this is inapplicable to active learning and the choice of acquisition function. Within the standardized workflow, the acquisition function is chosen among available heuristics a priori, and its success is observed only after the labelling budget is already exhausted. Picking a different acquisition function afterwards and starting again is not an option, as the labels have already been acquired. More importantly, none of the earlier studies reports a unique, consistently successful acquisition heuristic to the extent to stand out as the unique best choice.

We present a method to break this vicious circle by defining the acquisition function as a learning predictor and training it by reinforcement feedback collected from each labelling round. As active learning is a scarce data regime, we bootstrap from a well-known heuristic that filters the bulk of data points on which all heuristics would agree, and learn a policy to warp the top portion of this ranking in the most beneficial way for the character of a specific data distribution. Our system consists of a Bayesian neural net, the predictor, a bootstrap acquisition function, a probabilistic state definition, and another Bayesian policy network that can effectively incorporate this input distribution. We observe on three benchmark data sets that our method always manages to either come up with a new superior acquisition function or to adapt itself to the a priori unknown best performing heuristic for each specific data set.

## 4.1 BACKGROUND

In general, an active learning pipeline consists of three collaborating parts which we will refer to in this chapter as, predictor, guide, and oracle.

This chapter is based on and extends Haußmann et al. (2019).

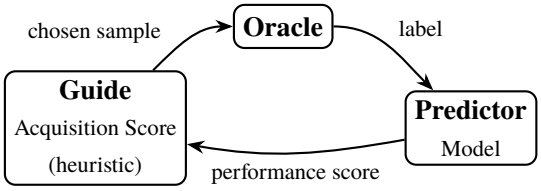



**Figure 4.1: Standard Active Learning pipeline.** The standard active learning pipeline is summarized as the interplay between three parts. An *oracle* provides a set of labeled data for a *predictor* (here a BNN) to learn on. It in turn provides predictive uncertainties to the *guide*, a usually fixed, hard-coded acquisition function, which in turn communicates to the oracle which points to label next, restarting the cycle.

The predictor is the model we are actually interested in. It is supposed to learn as much as possible from the available data to later perform well in practice. It learns from the given labelled data and then provides some performance score to the second part, the guide. Given the model's performance scores, the guide decides which data points should be labelled next requesting the corresponding target labels from an oracle. The oracle, usually a human expert, provides these and the circle starts again with the model being optimized on the now larger data set. This general setup is summarized in Figure 4.1.

Of these three, the oracle providing the labels, be it a human or another label source, can essentially be assumed as given and is thus irrelevant from a theoretical viewpoint. Of course, this is a simplification of the real world situation where a human expert might perform with different efficiency whether he/she needs to provide labels for a completely random order of data points, or whether it might be more efficient to provide and thus to request information on similar groups of data points. A guide, requesting labels, should take not only criteria into account that are theoretically optimal for the predictor model, but also incorporate such de facto relevant criteria. Throughout this chapter, we will ignore such details and assume an oracle that patiently provides labels for whatever data is requested with equal efficiency. Instead, we focus solely on the predictor and the guide and their relation and interplay with each other.

Contrary to the *tabula rasa ansatz* of the present deep learning age, state of the art in active learning has maintained hand-designed acquisition functions as the guides to rank the unlabeled sample space. However, different studies observe different acquisition functions to perform optimally for specific applications after evaluating various choices (Settles, 2012). The critical fact is that active learning is meant to address applications where data labelling is extremely costly. It is usually not possible to know the ideal acquisition function for a given application a priori. Once an acquisition function is chosen and active learning has been performed based on it, the labelling budget is already exhausted, leaving no possibility for another try with an alternative acquisition function. This limitation can only be circumvented by adapting this function to data during the active learning process, getting feedback from the previous labelling rounds' impact on the model fit. For real-world scenarios to which active

learning is applicable, learning also the acquisition function is then not only an option driven solely by practical concerns such as avoidance of handcrafting effort, but also an absolute necessity.

The acquisition functions in active learning are surrogate models that map a data point to a value that encodes the expected contribution of observing its label to model fit. The active learning setup's founding assumption is that evaluating the acquisition score of a data point is substantially cheaper than acquiring its ground-truth label. Hence, the acquisition functions are expected to be both computationally cheap and maximally accurate in detecting the sample space's most information-rich regions. Information-theoretic heuristics typically addresses these competing goals. Possibly the most frequently used acquisition heuristic is *Maximum Entropy Sampling*, which assigns the highest score to the data point for which the predictor reports the highest entropy (i.e. uncertainty). This criterion builds on the assumption that the most valuable data point is the one the model is maximally unfamiliar about. While being maximally intuitive, this method remains agnostic to exploiting knowledge from the current model fit about how much the new label can impact the model uncertainty. Another closely related heuristic with comparable reception, *Bayesian Active Learning by Disagreement (BALD)* (Houlsby et al., 2012), benefits from this additional information by maximizing the mutual information between the predictor output and the model parameters. A second major vein of research, which we will not pursue further in this chapter, approaches the active learning problem from a geometric instead of an uncertainty based perspective, incorporating the arrangement of the data points relative to each other in some feature space, for example via the selection of a core-set (Sener and Savarese, 2018).

None of the aforementioned heuristics has a theoretical superiority that is sufficient to rule out all other options. Maxent strives only to access unexplored data regions. BALD performs the same by also taking into account the expected effect of the newly explored label on the model parameters' uncertainty. While some studies argue in favour of BALD due to this additional information (Srinivas et al., 2012), others prefer to avoid this noisy estimate drawn from an under-trained model (Qiu et al., 2017).

In this chapter, we propose a data-driven method that alleviates the consequences of the unsolved acquisition function selection problem. As prediction uncertainty is an essential input to acquisition heuristics, we choose a deep Bayesian Neural Net (BNN) as our base predictor. To acquire high-quality estimates of prediction uncertainty with an acceptable computational cost, we devise a deterministic approximation scheme that can both effectively train a deep BNN and calculate its posterior predictive density following a chain of closed-form operations. Next, we incorporate all the probabilistic information provided by the BNN predictions into a novel state design, which brings about another full-scale probability distribution. This distribution is then fed into a second probabilistic policy network, which is trained by reinforcement feedback collected from every labelling round to inform the system about its

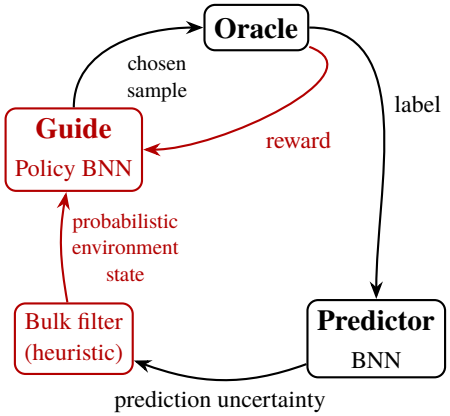



**Figure 4.2: The proposed pipeline.** We replace the fixed acquisition function with a policy BNN that learns with a probabilistic state and reinforcement feedback from the oracle how to optimally choose the next points (the differences from the general setup in Figure 4.1 are marked in red). It is thus able to adapt itself flexibly to the data set at hand.

current acquisition function's success. This feedback fine-tunes the acquisition function, bringing about improved performance in the subsequent labelling rounds. Figure 4.2 summarizes the high-level workflow of our method.

In the experiments, we evaluate our method on three common benchmark vision data sets from different domains and complexities: MNIST (LeCun et al., 1998) for images of handwritten digits, FashionMNIST (Xiao et al., 2017) for greyscale images of clothes, and CIFAR-10 (Krizhevsky and Hinton, 2009) for coloured natural images. In our experiments, we observe that the policy net is capable of inventing an acquisition function that outperforms all the handcrafted alternatives if the data distribution permits it. In the other cases, the policy net converges to the best-performing handcrafted choice, which varies across data sets and is unknown prior to the active learning experiment.

## 4.2 THE MODEL

As discussed above, our method consists of two major components: a predictor and a policy net guiding the predictor by learning a data set specific acquisition function. As the predictor, described in Section 4.2.1 we use a BNN, whose posterior predictive density we use to distil what we refer to as the system state. The policy net, a second BNN, takes this state as input to decide which data points to request labels for next. We describe this second part of the pipeline in Section 4.2.2. Since we introduce a reinforcement learning-based method for active learning, we refer to it as *Reinforced Active Learning (RAL)*.

Throughout this chapter, we rely for illustrative purposes on a central limit theorem based approach to marginalize the weights and improve both the predictive uncertainty and reduce the computational noise. In general, any approach to training a BNN, e.g. the Heavyside-Decomposition based VBP approach dis-

cussed at length in the last chapter, or any other predictor model which provides trustworthy enough predictive uncertainties could be used.

### 4.2.1 *The Predictor: A Bayesian Neural Network*

Let $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1}^N\}$ be a data set consisting of $N$ tuples of feature vectors $\boldsymbol{x}_n \in \mathbb{R}^m$ and labels $\boldsymbol{y}_n \in \{0,1\}^C$ for a $C$ dimensional binary output label. Parameterizing an arbitrary neural network $f(\cdot)$ with parameters $\boldsymbol{w}$ following some prior $p(\boldsymbol{w})$, we assume the following probabilistic model

$$\begin{aligned} \boldsymbol{w} &\sim p(\boldsymbol{w}), \\ \boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w} &\sim \prod_c^C \mathcal{B}er\big(y_c|\Phi(f_c(\boldsymbol{x};\boldsymbol{w}))\big), \end{aligned}$$

where $f_c$ is the $c$th output channel of the net, $\Phi(u) = \int_{-\infty}^{u} \mathcal{N}(x|0,1)dx$ is the normal cdf, and $\mathcal{B}er(\cdot|\cdot)$ is a Bernoulli distribution. Note that even thoughout the experiments we will only consider the multi-class case where $\sum_c y_c = 1$, this formulation treats each class independently, i.e. assumes a multi-label setup. This, as well as the switch to using the normal cdf instead of the more common logistic sigmoid, is a technical necessity we will discuss further below.

The calculation of the posterior predictive for a test pair $(\boldsymbol{x}^*, \boldsymbol{y}^*)$,

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}) p(\boldsymbol{w}|\mathcal{D}) \, \mathrm{d}\boldsymbol{w}$$

involves the calculation of the posterior distribution on the latent variables, which can be accomplished by Bayes rule

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{w}) p(\boldsymbol{w})}{\int p(\mathbf{Y}|\mathbf{X}, \boldsymbol{w}) p(\boldsymbol{w}) \, \mathrm{d}\boldsymbol{w}},$$

for $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $\mathbf{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$.

As this is intractable in general, we require approximate inference techniques. We aim for high-quality prediction uncertainties. A sampling-based approach is not practical for vision-scale applications where neural nets with a large parameter count are being used. Instead, we use variational inference. To keep the calculations maximally tractable while benefiting from stochastic $\boldsymbol{w}$, we formulate a normal mean-field variational posterior (which could be generalized):

$$q_\theta(\boldsymbol{w}) = \prod_i \mathcal{N}(w_i|\mu_i, \sigma_i^2), \tag{4.1}$$

where the tuple $(\mu_i, \sigma_i^2)$ represents the variational parameter set for weight $w_i$ of the network $f(\cdot)$ and $\theta = \{(\mu_i, \sigma_i^2)_i\}$.

Variational inference approximates the true intractable posterior by optimizing $\theta$ to minimize the Kullback-Leibler (KL) divergence between $q_\theta(\boldsymbol{w})$ and

$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{Y})$, which boils down to minimizing the negative evidence lower bound (see e.g. the Background chapter for details)

$$\mathcal{L}_{\text{class}}(\theta; \mathcal{D}) = -\sum_{n=1}^{N} \mathbb{E}_{q_\theta(\boldsymbol{w})} \left[\log p\big(\boldsymbol{y}_n | f(\boldsymbol{x}_n; \boldsymbol{w})\big)\right] + \text{KL}\left(q_\theta(\boldsymbol{w}) \parallel p(\boldsymbol{w})\right).$$

In this formula, the first term on the right-hand side maximizes the data fit (i.e. minimizes the reconstruction loss). The second term penalizes the posterior's divergence from the prior, inducing an Occam's razor principle.

The modeler has control on the model families of both $q_\theta(\boldsymbol{w})$ and $p(\boldsymbol{w})$. Hence, choosing the prior $p(\boldsymbol{w})$ suitably to the normally distributed $q_\theta(\boldsymbol{w})$ assures an analytically tractable solution for the $\text{KL}(\cdot \parallel \cdot)$ term. We use $p(w_i) = \mathcal{N}(w_i|0, \alpha^{-1})$ with a fixed precision $\alpha$. However, the data fit term involves a nonlinear neural net, which introduces difficulties for keeping the expectations tractable. A major issue is that we need to differentiate this term with respect to the variational parameters $\theta$, which appear in the density $q_\theta(\boldsymbol{w})$ with respect to which the expectation is taken. This problem is overcome by the reparameterization trick (Kingma and Welling, 2014), which re-formulates $q_\theta(\boldsymbol{w})$ as a sampling step from a parameter-free distribution and a deterministic variable transformation.

$$\mathcal{L}_{\text{class}}(\theta; \mathcal{D}) = -\sum_{n=1}^{N} \mathbb{E}_{p(\varepsilon)} \left[\log p\big(\boldsymbol{y}_n | f(\boldsymbol{x}_n; \boldsymbol{w} = \mu + \sigma\varepsilon)\big)\right] + \text{KL}\left(q(\boldsymbol{w}) \parallel p(\boldsymbol{w})\right),$$

where the parameters $\theta$ now appear only inside the expectation term. We could take the gradient of the loss with respect to them and approximate integral in the expectation by Monte Carlo sampling. Although this will provide an unbiased estimator of the exact gradient, this estimator will have prohibitively high variance due to distorting the global variables of a highly-parameterized system. Our remedy is to postpone sampling one step further.

Let the pre- and post-activation map of $j$th neuron of layer $l$ for data point $n$ be $f_{njl}$ and $h_{njl}$, respectively. We then have

$$w_{ijl} \sim \mathcal{N}(w_{ijl}|\mu_{ijl}, \sigma^2_{ijl}), \quad f_{njl} = \sum_{i=1}^{I_{l-1}} w_{ijl} h_{nil-1}, \tag{4.2}$$

as a repeating operation at every layer transition within a BNN.[1] As $h_{nil-1}$ is the sampling output of layer $l-1$, it is a deterministic input to layer $l$. Consequently, $f_{njl}$ is a weighted linear sum of $I_{l-1}$ independent normal random variables, which is another normal random variable with

$$f_{njl} \sim \mathcal{N}\left(f_{njl} \,\middle|\, \sum_{i=1}^{I_{l-1}} \mu_{ijl} h_{nil-1}, \sum_{i=1}^{I_{l-1}} \sigma^2_{ijl} h^2_{nil-1}\right). \tag{4.3}$$

1 The same line of reasoning directly applies to convolutional layers where the sum on $f_{njl}$ is performed in a sliding window fashion.

We now take separate samples for local variables, further reducing the estimator variance stemming from the Monte Carlo integration. This is known as Variational Dropout (Kingma et al., 2015), as the process performed for the expected log-likelihood term is equivalent to Gaussian dropout with rate $\sigma_{ijl}^2/\mu_{ijl}^2$ for weight $w_{ijl}$.

#### 4.2.1.1 *Fast Dropout and the CLT Trick*

Fast Dropout (Wang and Manning, 2013) has been introduced as a technique to perform Gaussian dropout without taking samples. The method builds on the observation that $f_{njl}$ as given in (4.2) is essentially a random variable consisting of a large sum of a provisionally large number of other random variables. This is a direct call to the Central Limit Theorem (CLT) that transforms the eventual distribution into a normal density, which one can trivially model by matching the first two moments

$$
\begin{aligned}
p(f_{njl}) &\approx \mathcal{N}(f_{njl}|\nu_{njl}, \lambda_{njl}^2), \\
\text{where} \quad \nu_{njl} &= \mathbb{E}\left[\sum_{i=1}^{I_{l-1}} w_{ijl} h_{nil-1}\right] = \sum_{i=1}^{I_{l-1}} \mathbb{E}\left[w_{ijl}\right] \mathbb{E}\left[h_{nil-1}\right], \\
\text{and} \quad \lambda_{njl}^2 &= \operatorname{var}\left[\sum_{i=1}^{I_{l-1}} w_{ijl} h_{nil-1}\right] \\
&= \sum_{i=1}^{I_{l-1}} \operatorname{var}\left[h_{nil-1}\right] \mathbb{E}\left[w_{ijl}\right]^2 + \operatorname{var}\left[w_{ijl}\right] \mathbb{E}\left[h_{nil-1}^2\right].
\end{aligned}
$$

Here, $\mathbb{E}\left[w_{ijl}\right] = \mu_{ijl}$ and $\operatorname{var}\left[w_{ijl}\right] = \sigma_{ijl}^2$, as determined in (4.1). We also require the first two moments over of the $h_{nil-1} = r(f_{nil-1})$, where $r(x) = \max\{0, x\}$ is the ReLU activation function, for which it suffices to solve

$$
\begin{aligned}
\mathbb{E}\left[h_{njl-1}\right] &= \int r(f_{njl-1}) \mathcal{N}(f_{njl-1}|\nu_{njl-1}, \lambda_{njl-1}^2)\, \mathrm{d}f_{njl-1}, \\
\mathbb{E}\left[h_{njl-1}^2\right] &= \int r(f_{njl-1})^2 \mathcal{N}(f_{njl-1}|\nu_{njl-1}, \lambda_{njl-1}^2)\, \mathrm{d}f_{njl-1}.
\end{aligned}
$$

For our choice of activation function, these two are analytically tractable (Frey and Hinton, 1999) and are given with $\Phi(\cdot)$, and $\phi(\cdot)$ referring to the standard normal cdf and pdf as

$$
\begin{aligned}
\mathbb{E}\left[h_{njl-1}\right] &= \nu_{njl-1}\Phi\left(\frac{\nu_{njl-1}}{\lambda_{njl-1}}\right) + \lambda_{njl-1}\phi\left(\frac{\nu_{njl-1}}{\lambda_{njl-1}}\right), \\
\mathbb{E}\left[h_{njl-1}^2\right] &= (\nu_{njl-1}^2 + \lambda_{njl-1}^2)\Phi\left(\frac{\nu_{njl-1}}{\lambda_{njl-1}}\right) + \nu_{njl-1}\lambda_{njl-1}\phi\left(\frac{\nu_{njl-1}}{\lambda_{njl-1}}\right).
\end{aligned}
$$

Note that $f_{njl-1}$ is either the linear activation of the input layer, a weighted sum of normals, hence another normal, or a hidden layer, which will then similarly follow CLT and therefore already be approximated as a normal. Hence, the above pattern repeats throughout the entire network, allowing a tight closed-form approximation of the analytical solution. We refer to this method as the *CLT trick* throughout this chapter. This approach will also be an integral part of our next chapter (Chapter 5), where we will discuss it in greater detail.

#### 4.2.1.2 *Closed-Form Uncertainty Estimation with BNNs*

Fast Dropout uses the aforementioned CLT trick only for implementing dropout. Here we extend the same method to perform variational inference by minimizing a deterministic loss, i.e. avoiding Monte Carlo sampling altogether. The state of the art in deep active learning relies on test-time dropout (Gal et al., 2017), which is computationally expensive. Speeding up this process requires parallel computing on the *end-product*, hence reflects additional costs on the user of the model not addressable at the production time. A thus-far overlooked aspect of the CLT trick is that it also allows closed-form calculation of the posterior predictive density if we choose the multi-label probit classification introduced above.

Once training is over, we get a factorized surrogate for our posterior. We can get this surrogate to approximate the true posterior predictive and get for a new observation $\boldsymbol{x}^*$ that

$$\begin{aligned} p(y_c^*|\boldsymbol{x}^*, \mathcal{D}) &\approx \int \mathcal{B}er(y_c^*|\Phi(f_c(\boldsymbol{x}^*; \boldsymbol{w})))q_\theta(\boldsymbol{w})\,\mathrm{d}\boldsymbol{w} \\ &\approx \int \mathcal{B}er(y_c^*|\Phi(f_c^*))\mathcal{N}\left(f_c^*|g_c^L(\boldsymbol{x}^*), h_c^L(\boldsymbol{x}^*)\right)\mathrm{d}f_c^* \\ &= \mathcal{B}er\left(y_c^*\middle|\Phi\left(\frac{g_c^L(\boldsymbol{x}^*)}{\sqrt{1+h_c^L(\boldsymbol{x}^*)}}\right)\right), \end{aligned} \tag{4.4}$$

where the functions $g_c^L(\boldsymbol{x}^*)$ and $h_c^L(\boldsymbol{x}^*)$ encode the cumulative map from the input layer to the moments of the top-most layer after repetitive application of the CLT trick across the layers. Once $g_c^L(\boldsymbol{x}^*)$ and $h_c^L(\boldsymbol{x}^*)$ are tightly approximated, one could also choose a categorical distribution as the likelihood and approximate the last integral via MC sampling. With $p(y_c^*|\boldsymbol{x}^*, \mathcal{D})$ being tightly approximated by an analytical calculation of a known distributional form, its high-order moments are readily available for downstream tasks, being active learning in our case.

In summary, the CLT based approach with the given likelihood gives us a sampling-free approximation to the posterior predictive we can use as a signal for the second part of the pipeline, the guide.

### 4.2.2 *The Guide: A Policy Net*

As opposed to the standard active learning pipeline, our method should be capable of adapting its acquisition scheme to the characteristics of individual data distributions. Differently from earlier work on data-driven label acquisition, it should be able to perform the adaptation *on the fly*, i.e. while the active learning labelling rounds take place. This adaptiveness is achieved within a reinforcement learning framework, where a policy net is trained by rewards observed from the environment. Below we provide the state, action, and reward definitions necessary to specify the proposed framework. We denote the collection of unlabeled points by $\mathcal{D}_u$ and the labelled ones by $\mathcal{D}_l$. The variables $N_u$ and $N_l$ denote the number of data points in each respective set.

*State*

STATE. In active learning, the label acquisition process, in general, takes place on the entire unlabeled sample set. However, a feasible reinforcement learning setup necessitates a somewhat condensed state representation to learn efficiently. To this end, we first rank the unlabeled sample set by an information-theoretic heuristic, namely the maximum entropy criterion. As this heuristic will assign similar scores to samples with similar features, consecutive samples in the ranking inevitably have a high correlation and simply picking the top $M$ points for the guide to choose among would essentially reduce it to a more complex maximum entropy heuristic, as there won't be much to choose from. To break the trend and enhance diversity, we, therefore, follow the ranking from the top and pick up every $K$th sample until we collect $M$ samples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$. We adopt the related term from the Markov Chain Monte Carlo sampling literature and refer to this process as *thinning*. Feeding these samples into our predictor BNN gives with our approximation to the predictive posterior (4.4), an estimate for each and distil the state of the unlabeled sample space in the following distributional form:

$$S \sim \prod_{c=1}^{C} \prod_{m=1}^{M} \mathcal{N}\left(s_{mc} \middle| g_c^L(\boldsymbol{x}_m), h_c^L(\boldsymbol{x}_m)\right),$$

where $g_c^L(\cdot)$ and $h_c^L(\cdot)$ are mean and variance of the activation the $c$th output neuron. That is instead of a deterministic state for the guide to work with, the state is given as a multivariate normal random variable.

*Action*

ACTION. At each labelling round, several data points are sampled from the set $\{\boldsymbol{x}_{i_1}, ..., \boldsymbol{x}_{i_M}\}$ as represented by the state $S$ according to the probabilities assigned on them by the present policy.

*Reward*

REWARD. The straight-forward and optimal reward would be the performance of the updated predictor on a separate validation set. This, however, clashes with the constraint imposed on us by the active learning scenario. The motivating assumption is that labels are valuable and scarce, so it is not feasible to construct a separate labelled validation set large enough to get a good guess of the desired test set performance for the policy net to calculate rewards. In our preliminary experiments, we have consistently observed that merging the validation set with the existing training set and performing active learning on the remaining sample set consistently provides a steeper learning curve than keeping a validation set for reward calculation. Hence, we abandon this option altogether. Instead, we propose a novel reward signal

$$R = R_{\text{improv}} + R_{\text{div}}$$

consisting of two components which we discuss next.

The first component $R_{\text{improv}}$ assesses the improvement in the chosen point's data fit once it has been labelled. From a Bayesian perspective, a principled measure of model this fit is the marginal likelihood. For a newly labelled pair

$(\boldsymbol{x}, \boldsymbol{y})$, we define this reward as the improvement in the marginal likelihood the predictor net assigns to the chosen point. This term is given as

$$R_{\text{improv}} = \prod_{c=1}^{C} \int \mathcal{B}er\big(y_c|\Phi(f_c(\boldsymbol{x};\boldsymbol{w}))\big) q_{\text{new}}(\boldsymbol{w})\,\mathrm{d}\theta - \prod_{c=1}^{C} \int \mathcal{B}er\big(y_c|\Phi(f_c(\boldsymbol{x};\boldsymbol{w}))\big) q_{\text{old}}(\boldsymbol{w})\,\mathrm{d}\theta,$$

where $q_{\text{old}}(\cdot), q_{\text{new}}(\cdot)$ are our respective variational posteriors before and after training with the new point, and each term can be computed as discussed before. The second component, $R_{\text{div}}$ encourages diversity across the selected labels throughout the whole labelling round:

$$R_{\text{div}} = \frac{\text{\#unique labels requested}}{\text{\#label requests in this episode}}.$$

*Policy net*

POLICY NET. The policy net $\pi(\cdot)$ is a second BNN parameterized by $\phi \sim p(\phi)$. Compared to the predictor net, which takes deterministic data points as input, the policy net takes the probabilistic state $S$, which follows a $CM$-dimensional normal distribution. Our CLT based BNN approach allows us to ensure that inputting such a stochastic input into our deterministic inference scheme is straightforward by using the first two moments of the state during the first moment-matching round. The output of the policy net, in turn, parameterizes an $M$ dimensional categorical distribution over possible actions. To benefit fully from the BNN character of the policy and to marginalize over the $\phi$ we again follow the approach we use for the classifier propagating the moments and first compute $M$ binary probabilities for taking action $\tilde{a}_m$ at time point $t$

$$p(\tilde{a}_m^t) = \int \mathcal{B}er\big(\tilde{a}_m^t|\Phi(\pi_m(S_t;\phi))\big) q(\phi)\,\mathrm{d}\phi, \tag{4.5}$$

and finally, normalize them for the categorical distribution $\mathcal{C}at(\cdot)$ such that the action is chosen from

$$A_t \sim \mathcal{C}at(\mathbf{a}^t), \qquad \text{where } a_m^t = \tilde{a}_m^t \Big/ \sum_j \tilde{a}_j^t.$$

ALGORITHM. As this chapter's primary objective is to demonstrate the applicability of the approach, we remain with a very standard, plain RL learning algorithm. In practical applications, one would rely on more advanced approaches. Here, we adopt the episodic version of the standard REINFORCE algorithm as proposed by Williams (1992) to train our policy net, with the adaptation of using a moving average over all the past rewards as the baseline to reduce the gradient noise. A labelling episode consists of choosing a sequence of points to be labelled (with a discount factor of $\gamma = 0.95$) after which the BNN is retrained, and the policy net takes one update step. We parameterize the policy $\pi_\phi(\cdot|S_t)$ itself by a neural network with parameters $\phi$. Our method iterates between labelling episodes, training the policy net $\pi$, and training the BNN $f$ until the labelling budget is exhausted. A pseudocode summary of the proposed method is given in Algorithm 1.

**Algorithm 1:** The RAL training procedure

**Input:** $\mathcal{D} = \{\mathcal{D}_u, \mathcal{D}_l\}$, labeling budget, state size $M$, policy $\pi_\phi$, net $f_\theta$, episode length $T$

```
// Train an initial net
```
train $f_\theta$ on $\mathcal{D}_l$ as described in Section 4.2.1
**while** *budget available* **do**
  `// The labeling episode`
  generate state distribution $S_0$ from $\mathcal{D}_u$
  **for** $t \in 1, ..., T$ **do**
    sample $A_t$ via $\pi_\phi(S_{t-1})$
    $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup \{\text{data point selected via } A_t\}$
    $\mathcal{D}_u \leftarrow \mathcal{D}_u \backslash \{\text{data point selected via } A_t\}$
    generate state distribution $S_t$ from $\mathcal{D}_u$
  **end**
  `// Update the agent and net`
  train $f_\theta$ on $\mathcal{D}_l$
  compute rewards $R_{\text{div}}, R_{\text{improv}}$ and returns $G_t$
  update $\phi$ via gradient descent on
  $G_t \nabla_\phi \left( \log \pi_\phi(A_t|S_t) + \text{KL}\left(q(\phi) \parallel p(\phi)\right)\right)$
**end**

## 4.3 EXPERIMENTS

As RAL is the first method to adapt its acquisition function while active learning occurs, its natural reference model is the standard active learning setup with a fixed acquisition heuristic. We choose the most established two information-theoretic heuristics: Maximum Entropy Sampling (Maxent) and BALD. Gal et al. (2017) already demonstrated that BNNs (in their case with fixed Bernoulli dropout) provide an improved signal to acquisition functions that can be used to improve upon using predictive uncertainty from deterministic nets. We will use our own BNN formulation for both RAL, and these baseline acquisition functions to give them access to the same closed-form predictive uncertainty and, therefore, ensure maximal comparability between our model and the baselines by having an identical architecture and training procedure for all methods. For Maxent one selects the point to be labeled next as the one that maximizes the predictive entropy,

$$\underset{(x,y)\in\mathcal{D}_u}{\arg\max}\, \text{H}\left(p(y|x,\mathcal{D}_l)\right), \qquad \text{where}$$

$$\text{H}\left(p(y|x,\mathcal{D}_l)\right) = -\sum_{c=1}^{C} p(y=c|x,\mathcal{D}_l) \log p(y=c|x,\mathcal{D}_l),$$

while BALD chooses the point that maximizes the expected reduction in posterior entropy, which is equivalent to choosing the point that maximizes

$$\underset{(x,y)\in\mathcal{D}_u}{\arg\max}\, H[p(y|x,\mathcal{D}_l)] - \mathbb{E}_{p(\boldsymbol{w}|\mathcal{D}_l)}\left[H[p(y|x,\boldsymbol{w})]\right].$$

We can compute maximum entropy as well as the first of the two BALD terms in closed form, while we calculate the second term of BALD via a sampling-based approach. We also include random sampling as a—on our kind of data rather competitive—baseline and evaluate the acquisition functions on three data sets to show the proposed method's adaptability to the problem at hand.

### 4.3.1 *Experimental Details*

To evaluate the performance of the proposed pipeline[2], we take as the predictor a standard LeNet5 sized model (two convolutional layers of 20, 50 channels and two linear layers of 500, 10 neurons respectively) and as the guide a policy net consisting of two layers with 500 hidden neurons in each layer. We use three different image classification data sets to simulate varying difficulty while keeping the architectures and hyperparameters fixed. MNIST serves as a simple data set containing greyscale digits. FashionMNIST remains greyscale but is more difficult as it consists of ten classes of clothing objects, and CIFAR-10 finally is a very difficult data set given the small classifier depth that requires the classification of coloured natural images.

The assumption of active learning that labels are scarce and expensive also entails the problem that a large separate validation set to evaluate and finetune hyperparameters is not feasible. In general, we followed the assumption that an AL setting does not allow us to spend valuable labelled data for hyper-parameter optimization so that they all remain fixed to the common defaults in the literature. Relying on a separate validation data set that could be used for hyper-parameter selection is sometimes done in the literature, but often relying on specially selected hyper-parameters and a small training set is dwarfed by the effect of reasonable defaults and adding the validation data to the training data allowing the model to learn a lot more, compensating for suboptimal hyperparameters by better parameters. Both neural nets are optimized via Adam (Kingma and Ba, 2015) using the defaults suggested in that paper. The BNN is trained for 30 epochs between labelling rounds, while the policy net gets one update step after each episode. To simulate the need to avoid costly retraining after each labelling episode, the predictor net is initialized to the learned parameters from the last episode, with a linearly decreasing learning rate after each episode. In each experiment, the state is constructed by ranking the unlabeled data points according to their predictive entropy and then taking every twentieth point until $M = 50$ points are reached. Since all three data sets consider a 10 class classification problem, the result is a 500 dimensional normal distribution as the input to the policy net. A labelling round consists of labelling 5 points, and the procedure is repeated until 400 labelled points are reached starting from an initial set of 50 random labelled points. While the set of initial points differed between repetitions, all four approaches started with the same random 50 points to avoid any influence due to that choice.

2 See `https://github.com/manuelhaussmann/ral` for a reference implementation.

**Table 4.1: Results.** The table gives the average and the final error on the test set in percent (± one standard deviation over five runs). AVERAGE ERROR gives the average over the whole labeling process, while FINAL ERROR reports the error after labeling 400 labeled points.

| | MNIST | FASHIONMNIST | CIFAR-10 |
|---|---|---|---|
| | AVERAGE ERROR | | |
| Random | 19.01±1.55 | 28.70±0.61 | 74.02±0.56 |
| Maxent | 19.81±2.10 | 30.43±1.48 | 74.52±0.72 |
| BALD | 17.22±1.33 | 30.93±0.97 | 74.21±1.28 |
| RAL | **16.38**±0.79 | **28.51**±0.71 | **73.32**±0.72 |
| | FINAL ERROR | | |
| Random | 10.41±2.28 | 24.64±0.48 | 69.78±0.69 |
| Maxent | 8.61±1.25 | 25.72±1.28 | 69.80±0.32 |
| BALD | 6.91±0.23 | 26.85±0.49 | 69.69±1.69 |
| RAL | **6.81**±0.99 | **23.69**±0.73 | **68.96**±1.03 |

### 4.3.2 *Results*

We summarize the results in Table 4.1 and show the development of the test set in Figure 4.3. RAL can learn to adapt itself to the data set at hand, always outperforming the baselines. On the simple MNIST using a random strategy for labelling points is rather ineffective as a lot of points are quickly trivial for the net. On FashionMNIST Maxent and BALD struggle a lot more, and on CIFAR-10 the task is so complicated for the small net that essentially any point can provide a good signal, with RAL only slightly outperforming the other methods. Note the rather large standard deviations between the runs indicating a strong dependence on the set of initial points and the stochasticity in the gradient descent updates.

Although RAL uses a thinned Maxent ranking to generate its state, it can improve upon that strategy in every case. An ablation study (see the end of this chapter) showed that while the thinning process can improve the plain Maxent in some settings if one were to use it as a fixed strategy, it is not sufficient to explain the final performance difference between RAL and Maxent.

Finally, it should be remarked that our central goal in these experiments is to evaluate the relative performance of RAL and the baselines and not the absolute performance. For a real-world application, one would use deeper architectures for the more complex data sets, incorporate pre-trained networks from similar labelled data sets, and use data augmentation to use the small labelled dataset maximally. Further benefits would come from using semi-supervised information, for example by assigning pseudo-labels to data points that the classifier assigns a high predictive certainty to (Wang et al., 2016). Such approaches would significantly improve the classifier performance for all models, but since

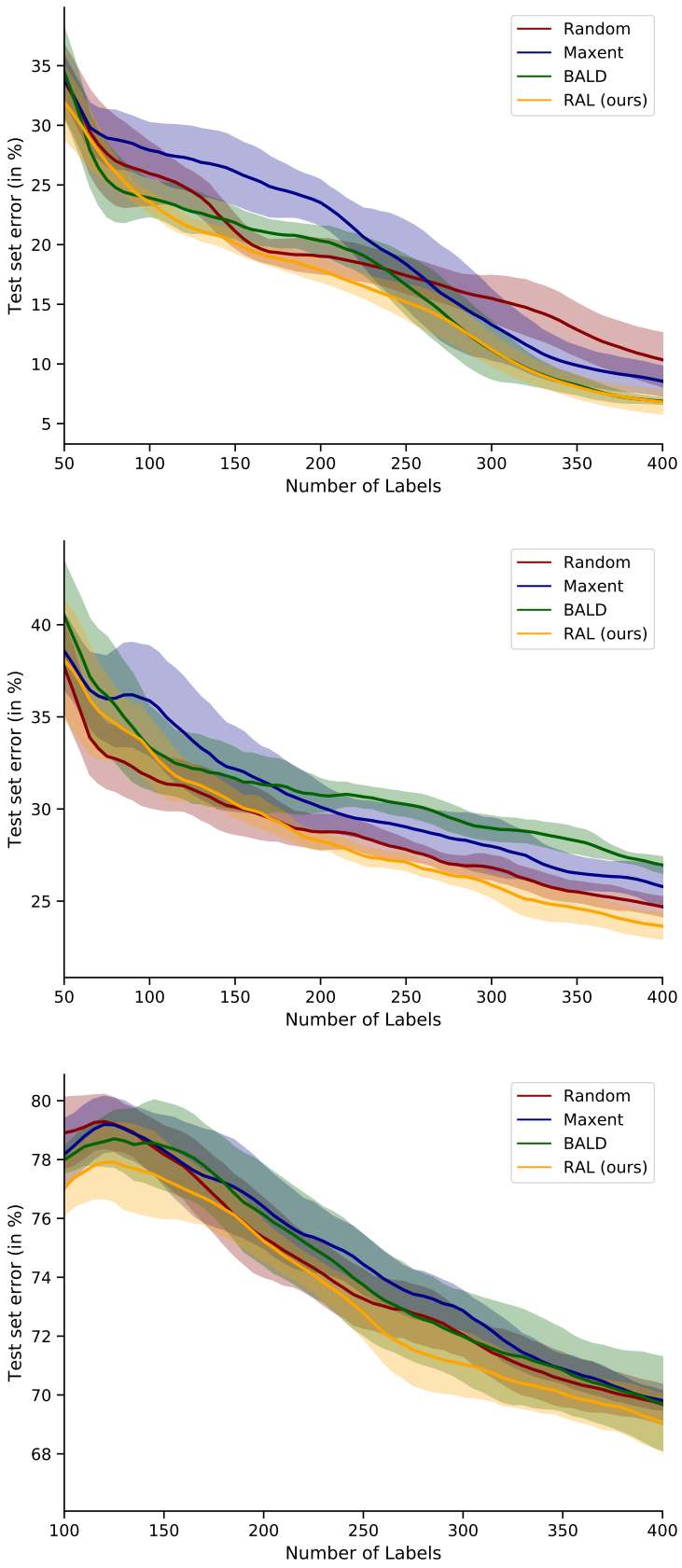



**Figure 4.3: Image Classification Results.** The development of the test set error throughout the labeling process. The plots are (from top to bottom) over MNIST, FashionMNIST, and CIFAR-10. In each the thick line is the smoothed average over five individual runs. The shading visualizes one standard deviation.

they would blur the respective acquisition function's contribution, we consciously ignore them here. REINFORCE owes its success to the bulk filtering step, which substantially facilitates the reinforcement learning problem by filtering out a large portion of the search space. The simplified problem can thus be improved within a small number of episodes. More interactions with the environment would certainly bring further improvement at the expense of increasing labelling cost. We are satisfied her weith presenting a proof-of-concept for the idea that can improve on the feedback-free AL even within limited interaction rounds. Further algorithmic improvements are worthwhile investigating as future work, such as applying TRPO (Schulman et al., 2015) or PPO (Schulman et al., 2017) in place of vanilla REINFORCE.

## 4.4 RELATED WORK

The gold standard in active learning methods has long remained to rely on hard-coded and hand-designed acquisition heuristics (see Settles (2012) for a recent review). A first extension is not limiting oneself to one heuristic, but to learn how to choose between multiple ones, for example by relying on a bandit algorithm (Baram et al., 2004; Hsu and Lin, 2015; Chu and Lin, 2016) or a Markov Decision Process (Ebert et al., 2012). However, this still suffers from the problem of being limited to existing heuristics.

A further step to gain more flexibility is to formulate the problem as a meta-learning task. The general idea (Fang et al., 2017; Konyushkova et al., 2017; Pang et al., 2018) is to use a set of labelled data sets to learn a general acquisition function that can either be applied as-is to the target data set or finetuned on a sufficiently similar set of labelled data. Our approach differs from those attempts because we learn the acquisition function based solely on the target data distribution while the data is labelled. If we take the scarcity of labels serious, we can't allow ourselves the luxury of a separate large validation set to adapt a general heuristic. Such a validation set also could not outperform the straight forward ablation study of allowing a simple hard-coded acquisition function that does not need a separate data set to instead combine the validation set with the labelled data to learn on. This is simply due to that as long as little labelled data is available, the gain from being able to learn from extra data tends to outweigh the benefit one would get by a complicated acquisition function, and as soon as data becomes more abundant, the effectiveness of any active learning method reduces sharply. We, therefore, discard such approaches from our comparative analysis. A similar related area is the field of metareasoning (Callaway et al., 2018), where an agent has to learn how to request based on a limited computational budget.

Alongside the sampling-based alternatives for BNN inference, which are already abundant and standardized (Blundell et al., 2015; Kingma et al., 2015; Gal and Ghahramani, 2016b; Louizos et al., 2017; Molchanov et al., 2017), deterministic inference techniques are also emerging. While direct adaptations of expectation propagation are the earliest of such methods (Hernández-Lobato

and Adams, 2015; Gast and Roth, 2018), they do not yet have a widespread reception due to their relative instability in training. This problem arises from the fact that expectation propagation does not provide any convergence guarantees. Hence an update might either improve or deteriorate the model fit on even the training data. Contrarily, variational inference maximizes a lower bound on the log-marginal likelihood. Early studies exist on deterministic variational inference of BNNs (Wu et al., 2019; Haußmann et al., 2020b). However, neither quantifies the uncertainty quality by using the posterior predictive of their models for a downstream application. Earlier work that performs active learning with BNNs does exist (Hernández-Lobato and Adams, 2015; Gal et al., 2017; Depeweg et al., 2018). However, all of these studies use hard-coded acquisition heuristics.

Our state construction method that forms a normal distribution from the posterior predictive of data points shortlisted by a bootstrap acquisition criterion is a novel idea for the active learning setting. It has strong links to model-based reinforcement learning methods that propagate uncertainties through one-step predictors along the time axis (Deisenroth and Rasmussen, 2011).

## 4.5 CONCLUSION

We discussed in this chapter the introduction of a new reinforcement-based method for labelling criterion learning. It can learn how to choose points parallel to the labelling process itself instead of requiring large already labelled subsets to learn on in an off-line setting beforehand. We achieve this by formulating the classification net, the policy net as well as the state probabilistically. We demonstrate its ability to adapt to various qualitatively different data set situations performing similar to or even outperforming handcrafted heuristics. In future work, we plan to extend the policy net with a density estimator that models the input data distribution to take the underlying geometry into account, making it less dependent on the quality of the probabilities.

## 4.6 FURTHER DETAILS

We close this discussion on active learning with a series of ablations. The first of these targets the question of the state generation. How much of the model performance is due to the entropy-based preprocessing and thinning we discussed? The second and third target the CLT approach itself. Yes, it is theoretically pleasing, and the sample-free structure should give us cleaner gradients and a better uncertainty signal, but does it? We first evaluate its influence on the convergence speed and then the quality of the predictive uncertainty with respect to the different acquisition functions.

Finally, we provide the remaining hyperparameters for the experiments that were not discussed in the main part.

**Table 4.2:** This table gives the corresponding results to Figure 4.4 giving the mean ± one standard deviation averaged over five runs.

| **FashionMNIST** | AVG. ERROR | FINAL ERROR |
|---|---|---|
| Random | 28.70±0.61 | 24.64±0.48 |
| Maxent | 30.43±1.48 | 25.72±1.28 |
| Maxent_thinned | 29.81±1.59 | 24.53±1.58 |
| RAL_nograd | 28.57±0.54 | 24.75±0.67 |
| Reinit RAL | 28.90±0.10 | 24.12±0.99 |
| RAL | **28.51**±0.71 | **23.69**±0.73 |

### 4.6.1 *Ablation Study on State Construction and the Guide*

A question that arises from our results is whether the improvements are actually due to the policy net adapting itself to the current data set or whether the credit should go to the bootstrapping and thinning done in the state definition. To evaluate this, we run two different ablation experiments on FashionMNIST:

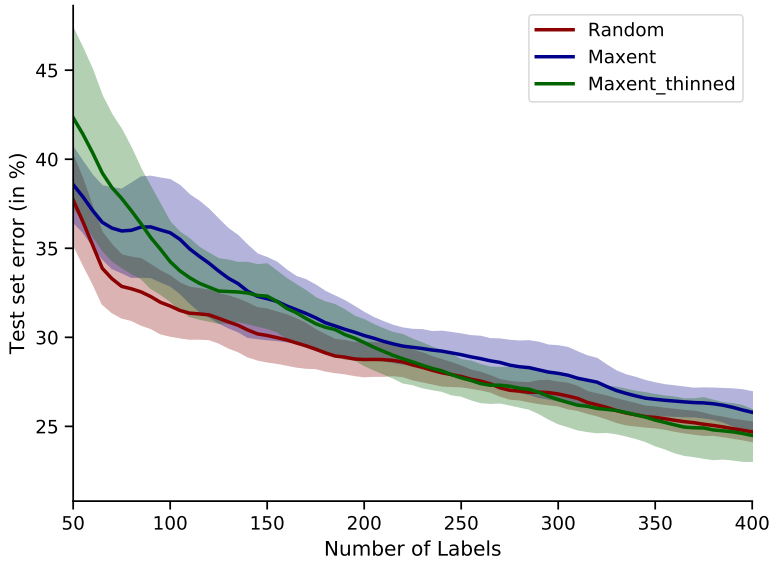
1. We consider applying the same thinning approach we use in the state construction to the Maxent criterion training. Similar to our state derivation, we pick every twentieth point as ranked by Maxent until reaching 50 points and randomly pick five of those. One can see this as relaxing the pure exploitation of Maxent by encouraging more exploration. We refer to this variation as *Maxent_thinned* and show that it improves upon Maxent. However, it still does not reach RAL performance. *Maxent_thinned*

2. To further explore the influence of the policy net, we consider three variations. (i) The original RAL model as proposed in the main discussion; (ii) A variation we refer to as *RAL_nograd*, where we fix the policy at a random initialization, leaving it to essentially pick randomly among the constructed state; (iii) *Reinit RAL* where we reinitialize it after every labelling episode, to explore how fast it converges and whether this helps it to adapt to new datapoints without getting stuck in local minima. RAL can always perform at least as good or better than this version, indicating that it jumps over suboptimal local minima while fixing the gradients returns the model's performance to random sampling as expected. *RAL_nograd* *Reininit RAL*

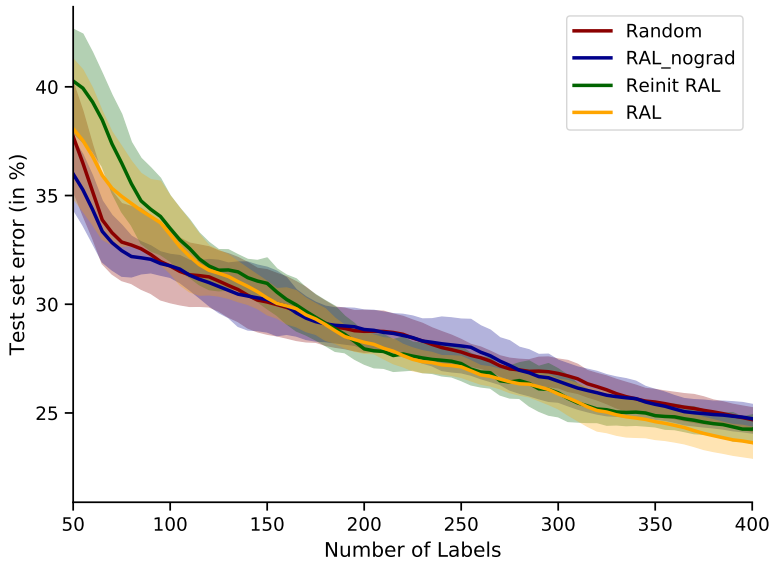
We summarize theis ablation experiment visually in Figure 4.4. Table 4.2 gives the corresponding average and final error.

### 4.6.2 *Convergence and Classification Accuracy*

To evaluate the performance and convergence speed of a dropout trained BNN with our proposed CLT-based BNN, we compare the following variations:




**(a)**




**(b)**

**Figure 4.4: Ablation study. (a)** gives the comparison of classical Maximum Entropy vs the thinned version. **(b)** summarizes the ablation experiments on RAL. The bold line in each gives the mean and the shaded area visualizes one standard deviation over five runs.

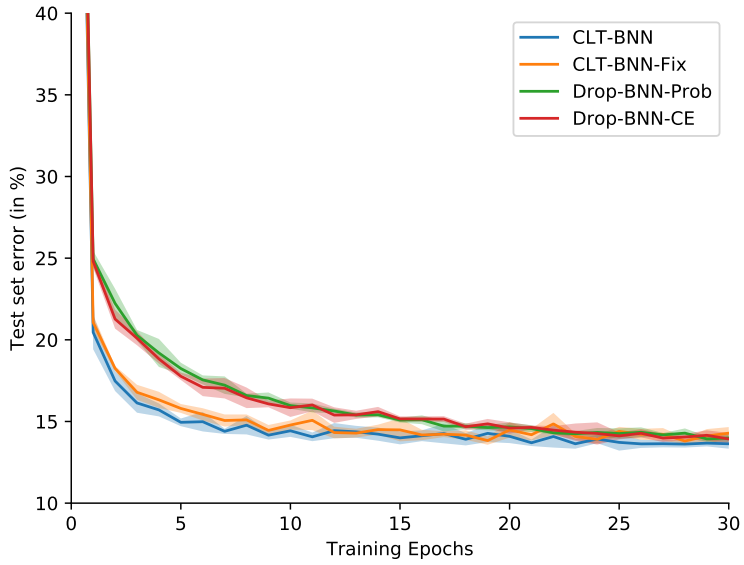



**Figure 4.5: FashionMNIST Convergence Comparison.** An ablation comparing the performance of dropout based nets with our proposed model. The filled areas give $\pm$ one standard deviation over five runs.

i) **Drop-BNN-CE** a deterministic network with binary dropout ($p = 0.5$) between all hidden layers and a categorical cross-entropy loss;

ii) **Drop-BNN-Prob** which uses the same binary Probit likelihood as our proposed model;

iii) **CLT-BNN-Fix** which is our proposed CLT based network, but with fixed variance parameters, to mimic a Gaussian dropout with an equivalent rate as the binary dropout used for i);

iv) **CLT-BNN** which is our proposed CLT based network with learnable variance parameters, i.e. the one we use in the main experiments.

We evaluate them on our dataset of intermediate difficulty, FashionMNIST. The results are the average over five runs of random initialization on a random subset of 10000 data points (batch size 64), the latter being identical for each method in a single run, to have both data as well as parameter variation. Prior precision for the BNNs is balanced with the weight decay of the Dropout-BNNs.

We report the perforamce in Figure 4.5. While all four methods eventually converge to the same performance, the two *CLT-BNN* variations converge a lot faster, with our proposed CLT-BNN being the most efficient.

### 4.6.3 *Quality of Uncertainty Estimation*

To compare our proposed model propagating the variances in closed form, instead of the sample-based dropout approach, we repeat the active learning

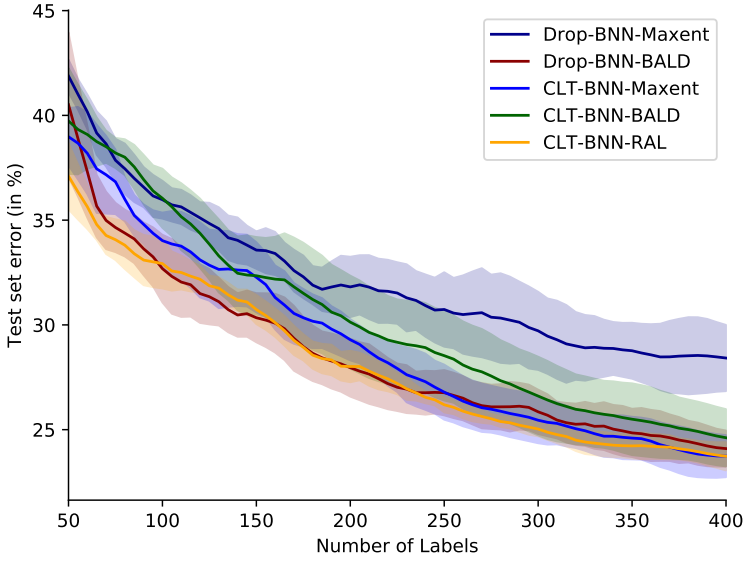



**Figure 4.6: FashionMNIST Uncertainty Comparison.** An ablation comparing the performance of dropout based nets with our proposed model. The filled areas give $\pm$ one standard deviation over five runs.

experiment on FashionMNIST in the following variation. Drop-BNN gets 12 samples to estimate Maxent and BALD. To further show some variation, we switch two of the primary hyperparameters. We increase the initial learning rate to $10^{-3}$ and raise the batch-size to 64 to compare the method's robustness to such kind of changes. We report the results in Figure 4.6. RAL remains the overall best-performing approach, while the difference in predictive accuracy is most evident in the performance difference of *Drop-BNN-Maxent* and *CLT-BNN-Maxent*, where the latter's closed-form approach greatly improve upon a sample-based dropout one.

## 4.7 FURTHER DETAILS ON THE EXPERIMENTS

CLASSIFIER ARCHITECTURE. Throughout the experiments, the structure we use for the BNN classifier closely follows the LeNet architecture described in the Caffe library[3], with strided convolutions replacing the two max-pooling layers. See Table 4.3 for the details of the MNIST/FashionMNIST experiments and Table 4.4 for the CIFAR-10 experiment (the latter gets a larger set of weights and filters). Each layer, except for the last, is followed by a ReLU activation, propagating the moments as described in the main discussion. The prior precision of their weights is assumed to be $\alpha = 10$ for all experiments.

3 `https://github.com/BVLC/caffe/blob/master/examples/mnist/lenet_train_test.prototxt`

**Table 4.3:** Classifier architecture for MNIST/FashionMNIST

| MNIST/FashionMNIST |
|---|
| $(1, 28, 28)$ dim input |
| Convolution $(5 \times 5)$ with 20 channels, stride=2 |
| Convolution $(5 \times 5)$ with 50 channels, stride=2 |
| Linear Layer (800, 500) |
| Linear Layer (500, 10) |

**Table 4.4:** Classifier architecture for CIFAR-10.

| CIFAR-10 |
|---|
| $(3, 32, 32)$ dim input |
| Convolution $(5 \times 5)$ with 192 channels, stride=2 |
| Convolution $(5 \times 5)$ with 192 channels, stride=2 |
| Linear Layer (4800, 1000) |
| Linear Layer (1000, 10) |

STATE CONSTRUCTION. The state is constructed by first ranking the unlabeled points by their predictive entropy with respect to the current variational posterior and taking the penultimate moments (i. e.pre-Probit) of every twentieth point until reaching 50 points. The state is then their concatenation, which means it is a 500 dimensional multivariate normal distribution with diagonal covariance structure.

POLICY NET ARCHITECTURE. The policy net consists of a BNN with two linear layers, with a ReLU activation between them following the derivations in the main discussion. Its architecture is summarized in Table 4.5. The prior precision on the weights is $\alpha = 100$.

**Table 4.5:** Policy net architecture.

| Policy Net |
|---|
| $(500, 2)$ dim (mean, variance) input |
| Linear Layer (500, 256) |
| Linear Layer (256, 50) |

RETURNS AND BASELINES. The returns are computed with a discount factor of $\gamma = 0.95$ as $G_i = \sum_{j=1}^{T-i} \gamma^j R_{i+j}$, and the policy gradient uses a moving average baseline.

OPTIMIZATION. Both, the classifier BNN and the policy net, are optimized using the Adam optimization algorithm introduced by Kingma and Ba (2015) with their proposed hyperparameters, which are $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon =$

$10^{-8}$. The learning rate for the classifier starts as $\lambda_{\text{cl}} = 10^{-4}$ and is linearly reduced after each labeling episode. The policy net has a fixed learning rate of $\lambda_{\text{pol}} = 10^{-3}$ throughout an experiment.

TRAINING PROCEDURE. The training procedure is summarized in Algorithm 1 above. A labeling round consists of labeling 5 new points, after which the policy net gets one update step and the underlying classifier gets 30 epochs (batchsize 10) worth of gradient descent steps.

# 5

# BAYESIAN EVIDENTIAL DEEP LEARNING WITH PAC-BAYES REGULARIZATION

As we have seen in the last chapters, the probabilistic weight structure of BNNs allows them to express complex uncertainty structures. However, as we have also seen, the exact inference of such a highly nonlinear system is analytically intractable and extremely hard to approximate with high precision. As we have discussed, research on BNNs thus far focused primarily on improving approximate inference techniques in precision and computational cost (Hernández-Lobato and Adams, 2015; Kingma et al., 2015; Louizos and Welling, 2017). All these prior attempts take the posterior inference of global parameters as given and develop their approximation based on it.

A newly emerging alternative approach is direct predictive distribution modelling. This approach proposes devising a highly expressive predictive distribution with several free parameters. These parameters are then fit to the data via maximum likelihood estimation. This way, observations are used to train the end product of interest directly: the predictive distribution, bypassing the need for the intractable posterior inference step. Some existing methods model the predictive distribution via a stochastic process parameterized as a neural net (Garnelo et al., 2018a). Others introduce local priors on the likelihood activations, integrate them out and train the hyperparameters of the resultant marginal (Malinin and Gales, 2018; Sensoy et al., 2018).

In this chapter, we first discuss how we directly target the posterior predictive in a Bayesian Neural Network (BNN). We assume independent local weight random variables controlling the BNN for each input/target pair of data points, which share common hyperparameters. We marginalize these data-point specific weights of our network and perform training via type-II maximum likelihood/empirical Bayes on the prior hyperparameters. This analytically intractable marginalization is approximated using the Central Limit Theorem (CLT), which we already saw used in a variational inference interpretation in the last chapters. Differently from earlier weight marginalization approaches that assign global weight distributions on infinitely many neurons that recover a Gaussian Process (Neal, 1995; Lee et al., 2018a; Garriga-Alonso et al., 2019), our formulation maintains finitely many hidden units per layer and assigns them individual weight distributions. Due to the weight marginalization per data-point treatment, the BNN scales training linearly with the data set size. Adopting empirical Bayes training on this simplified setup, our method avoids

This chapter builds on and extends Haußmann et al. (2020a). In that publication, Sebastian Gerwinn contributed part of the PAC-Bayes discussion and the adaptation of the KL inversion.

the explicit approximation of a highly nonlinear and intractable weight posterior yet can improve the quality of uncertainty estimations.

Given that training framework, we extend it by including the second approach, i.e. the inclusion of a "local prior" that is to be optimized over. For that, we combine it with a state of the art of that latter approach, Evidential Deep Learning (EDL) (Sensoy et al., 2018), due to its technical simplicity (e.g. it does not require access to out of distribution data during training as for example the otherwise related method of Malinin and Gales (2018) requires) and observed effects on experimental data. EDL places a Dirichlet prior on the class assignment probabilities of a classifier and parameterizes the Dirichlet strengths of this prior with a neural net. While demonstrating substantial improvements in out-of-distribution (OOD) detection and adversarial robustness, EDL is not capable of decomposing epistemic and aleatoric uncertainties. We propose an efficient and effective method that extends EDL to BNNs, equipping it with more advanced uncertainty quantification and decomposition capabilities. The advantages of such a BNN with data-point specific marginalization comes at the expense of a major drawback. As the number of hyperparameters in a weight-marginalized BNN grows proportionally to the number of synaptic connections and hence maximizing the marginal likelihood with respect to such a large number of hyperparameters is prone to overfitting (Bauer et al., 2016). Since the weight variables are marginalized out and their hyperparameters of the weight prior are set via optimization, the model can no longer incorporate regularizing knowledge other than the parametric form of the prior distribution (e.g. normal with mean and variance as free parameters). We address this drawback by deriving a Probably Approximately Correct Bayes (PAC-Bayes) (McAllester, 1999, 2003) bound that contains the marginal likelihood as its empirical risk term. Minimization of this PAC-Bayes bound automatically balances the fit to the data and deviation from a prior regularizing hypothesis.

We refer to the model as *Bayesian Evidential Deep Learning (BEDL)* and evaluate it on various standard regression, classification, and out-of-distribution detection benchmarks against state-of-the-art approximate posterior inference based BNN training approaches. We observe that our method provides competitive prediction accuracy and better uncertainty estimation scores than those baselines.

To summarize the structure of this chapter. We first discuss in Section 5.1 how to adapt the CLT approach to optimizing the posterior predictive directly. Section 5.2 then gives a background on the evidential deep learning setup by Sensoy et al. (2018) and how to combine the two. We close the theoretical discussion in Section 5.3 with a principled way to regularize the joint model via a PAC-Bayes bound, evaluating it in Section 5.6. See the further details section at the end of this chapter for some extended technical discussions that would distract from the main storyline. Figure 5.1 gives a high-level conceptual summary of the approach.

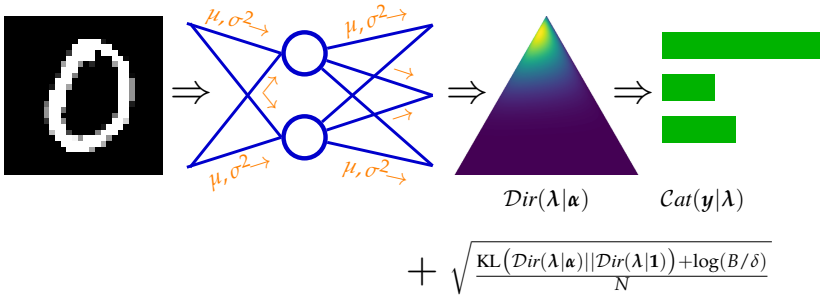



**Figure 5.1: The proposed model for Bayesian Evidential Deep Learning.** The orange arrows on the Bayesian Neural Network indicate the progressive flow of the moments during the weight integration phase. The output layer of the BNN determines the Dirichlet strengths of the prior on the class probability masses. We train this model using empirical Bayes/type-II maximum likelihood supported by a complexity penalty term derived from PAC-Bayesian principles.

## 5.1 BAYESIAN LOCAL NEURAL NETWORKS

Given a data set $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1}^N\}$ consisting of $N$ pairs of input $\boldsymbol{x}_n$ and target $\boldsymbol{y}_n$, parameterizing the likelihood by a BNN $f(\cdot; \boldsymbol{w})$ with random variables $\boldsymbol{w}$ as the weights results in the generative model

$$\begin{aligned} \boldsymbol{w} &\sim p(\boldsymbol{w}), \\ \boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w} &\sim p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}_n), \qquad \forall n \end{aligned} \tag{5.1}$$

which we are familiar with by now from the earlier chapters. As usual, the likelihood $p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w})$ depends on the target domain, for example for regression one would have $p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{y}_n|f(\boldsymbol{x}_n; \boldsymbol{w}), \beta^{-1}\right)$, with precision $\beta$, and some prior $p(\boldsymbol{w})$.

As discussed in the last chapters, the canonical variational inference based approach (Blundell et al., 2015; Kingma et al., 2015; Gal and Ghahramani, 2016a; Louizos and Welling, 2017) would now be to maximize the evidential lower bound

$$\log p(\mathcal{D}) \geq \mathbb{E}_{q_\theta(\boldsymbol{w})}\left[\log p(\mathcal{D}|\boldsymbol{w})\right] - \mathrm{KL}\left(q_\theta(\boldsymbol{w}) \parallel p(\boldsymbol{w})\right), \tag{5.2}$$

with respect to the parameters controlling the variational approximation to the posterior $q_\theta(\boldsymbol{w})$. As discussed in the corresponding section of the background chapter and as we have seen in the earlier BNN models, in a deep learning setting, contrary to more classical models, we are usually not interested in the posterior in itself, but only insofar as it is a necessary step on the path towards the posterior predictive

$$\begin{aligned} p(\boldsymbol{y}^*|\mathcal{D}, \boldsymbol{x}^*) &= \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}) p(\boldsymbol{w}|\mathcal{D})\, \mathrm{d}\boldsymbol{w} \\ &\approx \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}) q_\theta(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}, \end{aligned}$$

for some test point $\boldsymbol{x}^*$. This raises the question of how one could directly tackle the posterior predictive? This distribution is exactly the left-hand side of the evidence lower bound on the training data, i.e. the marginal likelihood $p(\mathcal{D})$ also known as evidence.

A well-established approach is to use this quantity for model comparison or selection. It suggests marginalizing out all latent variables, comparing the marginal likelihoods of all possible hypotheses, and choosing the one providing the highest response (Kass and Raftery, 1995). If we assume that the prior is not parameter-free but depends on some hyperparameters, e.g. a prior precision in the case of a normal prior, maximizing the evidence with respect to these hyperparameters is common practice, instead of introducing hyperpriors over these parameters and inferring posteriors over them (which would also just postpone the question as to the choice of their parameters controlling these hyperpriors). Referred to as type-II maximum likelihood or empirical Bayes (Bishop, 2006; Efron, 2012), this technique is fundamental for fitting non-parametric models such as Gaussian Processes (see e.g. Chapter 5 in Rasmussen and Williams, 2006).

We could then consider the following variation on the model above

$$\begin{aligned} \boldsymbol{w} &\sim p_\theta(\boldsymbol{w}), \\ \boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w} &\sim p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}_n), \qquad \forall n. \end{aligned} \tag{5.3}$$

Instead of a parameter-free prior $p(\boldsymbol{w})$ we switch to a parametric prior $p_\theta(\boldsymbol{w})$ and learn its hyperparameters via a type-II maximum likelihood approach

$$\arg\max_\theta \log p_\theta(\mathcal{D}).$$

This idea avoids the variational posterior but poses three problems.

First, for a model such as a BNN, the true posterior will by highly multimodal, a fact that we cannot reproduce by this approach, given the usually factorized priors. We ignore this problem for now insofar as it shared with the common mean-field variational posterior approach, which is also a very naive approximation to the true posterior. In both cases, the approximation is primarily motivated out of technical necessity to keep everything scalable.

The second problem is more profound as it is one where the suggested approach differs from the variational inference one. That is, the ELBO in (5.2) decomposes into a data-fit term and a regularizing term using a fixed prior. The proposition is regularized only through the fixed factorization and the marginalization, loosing both the regularizing term of a fixed prior and its function to include real prior knowledge in the fixed parameters. We will discuss a principled approach to tackle this problem in Section 5.3.

The third problem relates to this data fit term. Variational inference considers the expected log-likelihood

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

while the proposal can be understood as instead considering the logarithm of the expected likelihood

$$\log p(\mathcal{D}) = \log \mathbb{E}_{p_\theta(\boldsymbol{w})} \left[ p(\mathcal{D}|\boldsymbol{w}) \right] \geq \mathbb{E}_{p_\theta(\boldsymbol{w})} \left[ \log p(\mathcal{D}|\boldsymbol{w}) \right],$$

where the inequality holds via Jensen's inequality. Assuming that both the prior and the variational posterior are mean-field versions of the same distributions, optimizing this lower bound would revert to optimizing the evidence lower bound after dropping the Kullback-Leibler term. Focusing, however, on the left-hand side, the is primarily due to the large data sets considered in a deep learning setup. The expectation of the log-likelihood decomposes into

$$\begin{aligned}
\mathbb{E}_{q_\theta(\boldsymbol{w})} \left[ \log p(\mathcal{D}|\boldsymbol{w}) \right] &= \mathbb{E}_{q_\theta(\boldsymbol{w})} \left[ \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}) \right] \\
&= \sum_{n=1}^{N} \mathbb{E}_{q_\theta(\boldsymbol{w})} \left[ \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}) \right] \\
&\approx \frac{M}{N} \sum_{m=1}^{M} \mathbb{E}_{q_\theta(\boldsymbol{w})} \left[ \log p(\boldsymbol{y}_m|\boldsymbol{x}_m, \boldsymbol{w}) \right],
\end{aligned}$$

allowing in the last step for a mini-batch approximation which will be necessary for efficient training in practice. The log of the expectation however does not allow for this decomposition

$$\log p(\mathcal{D}) = \log \int \prod_{n=1}^{N} p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}) p_\theta(\boldsymbol{w}) \, \mathrm{d}\boldsymbol{w},$$

due to the shared weights. We therefore formulate the generative model as *BLNN*

$$\begin{aligned}
\boldsymbol{w}_n &\sim p_\theta(\boldsymbol{w}_n), \qquad \forall n \\
\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w} &\sim p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}_n), \qquad \forall n,
\end{aligned} \tag{5.4}$$

that is, we have a local set of weights for each data pair, sharing a common set of hyper-parameters $\theta$. As this modification implies a collection of only local latent variables, we refer to the resultant probabilistic model as a *Bayesian Local Neural Net (BLNN)*.

Introducing an independent $\boldsymbol{w}_n$ for each pair $(\boldsymbol{x}_n, \boldsymbol{y}_n)$ leads to a marginal likelihood formulation corresponding to a sum of $N$ independent marginal likelihoods, giving the desired decomposition structure

$$\begin{aligned}
\log p_\theta(\mathcal{D}) &= \log \int \prod_{n=1}^{N} p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}_n) p_\theta(\boldsymbol{w}_n) \, \mathrm{d}\boldsymbol{w}_1 \cdots \mathrm{d}\boldsymbol{w}_N \\
&= \sum_{n=1}^{N} \log \int p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}_n) p_\theta(\boldsymbol{w}_n) \, \mathrm{d}\boldsymbol{w}_n \\
&= \sum_{n=1}^{N} \log p_\theta(\boldsymbol{y}_n|\boldsymbol{x}_n) \approx \frac{N}{M} \sum_{m=1}^{M} \log p_\theta(\boldsymbol{y}_m|\boldsymbol{x}_m)
\end{aligned} \tag{5.5}$$

for a mini batch of size $M$.

The density function of the marginal likelihood of a training data point is identical to the posterior predictive density for new test observation $\boldsymbol{x}^*$. Hence, an analytic approximation developed for training is directly applicable at test time. The independence assumption across data points has various benefits. Provided that the integrals in (5.5) are analytically tractable, the computational complexity scales linearly with the training set size.

### 5.1.1 *Analytic Marginalization of Local Weights with Moment Matching*

Marginalizing out the local weights $\boldsymbol{w}_n$ in each term in (5.5) is an intractable problem due to the highly nonlinear neural net appearing in the likelihood $p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w}_n)$. However, we can follow the approach discussed in the last chapter and marginalize the weights approximately by recursive moment matching resorting to the Central Limit Theorem (CLT). This technique has previously been used in BNNs for other purposes, such as expectation propagation (Hernández-Lobato and Adams, 2015; Ghosh et al., 2016), fast dropout (Wang and Manning, 2013), and variational inference (Haußmann et al., 2019; Wu et al., 2019).

We employ the same technique for marginalizing out the weights of the BLNN. For a single data point and the $l$-th hidden fully-connected layer[1] consisting of $K$ units with an arbitrary activation function $a(\cdot)$, the post-activation layer output is given as $\boldsymbol{h}^l = a(\boldsymbol{f}^l)$, where $\boldsymbol{f}^l = \boldsymbol{W}^l \boldsymbol{h}^{l-1}$. The $j$-th pre-activation output $f_j^l$ is a sum of $K$ terms $f_j^l = \sum_k w_{jk}^l h_k^{l-1}$, which allows us to assume it to be normally distributed via the CLT due to the independence of the individual $w_{jk}^l$ and $h_k^{l-1}$ terms. The mean and the variance of this random variable are

$$\mathbb{E}\left[f_j^l\right] = \sum_{k=1}^{K} \mathbb{E}\left[w_{jk}^l\right] \mathbb{E}\left[h_k^{l-1}\right],$$
$$\operatorname{var}\left[f_j^l\right] = \sum_{k=1}^{K} \mathbb{E}\left[(w_{jk}^l)^2\right] \operatorname{var}\left[h_k^{l-1}\right] + \operatorname{var}\left[w_{jk}^l\right] \left(\mathbb{E}\left[h_k^{l-1}\right]\right)^2,$$

where we drop any potential covariance structure between the outputs of a layer. Although computing and propagating full covariance matrices across the layers would be mathematically feasible subject to further approximations, the computational costs (both memory and runtime wise) quickly become too large for such an approach to be tractable for anything but the tiniest networks.

The mean and the variance of the weights are readily available via the distributions $p_\theta(\boldsymbol{w}_n)$, leaving only the first two moments of $h_k^{l-1}$ undetermined. For common activations such as the ReLU, $a(h_k^{l-1}) = \max(0, h_k^{l-1})$, which we will rely on in this work, closed-form solutions to these moments are tractable (Frey and Hinton, 1999) given the moments of the pre-activations of the previous layer $\boldsymbol{f}^{l-1}$ (see Section 5.8.1 for details). This gives a recursive scheme terminating at the input layer, since we have $f_j^1 = \sum_k w_{hj}^1 x_k$. As $x_k$ is

1 Convolutional layers follow analogously.

usually a constant, its first moment is itself and the second is zero. This gives us that for the first layer pre-activations we have

$$\mathbb{E}\left[f_j^1\right] = \sum_{k=1}^{K} \mathbb{E}[w_{jk}^1] x_k,$$
$$\text{var}\left[f_j^1\right] = \sum_{k=1}^{K} \text{var}[w_{jk}^l] x_k^2,$$

completing the full recipe of how all weights of a BNN can be recursively integrated out from bottom to top, subject to a tight approximation. Scenarios with stochastic input $\boldsymbol{x} \sim p(\boldsymbol{x})$ typically entail controllable assumptions on $p(\boldsymbol{x})$. The equations above remain intact after adding an expectation operator around $x_k$ and $x_k^2$, readily available for any explicitly defined $p(\boldsymbol{x})$. Contrarily to the case in GPs, stochastic inputs can so be trivially adapted into this framework, greatly simplifying the math for uncertainty-sensitive setups, such as PILCO (Deisenroth et al., 2015).

For a net with $L$ layers, the outcome of recursive moment matching is a distribution over the final latent hidden layer $\boldsymbol{f}_n^L \sim \mathcal{N}(\boldsymbol{f}_n^L | \boldsymbol{m}_n, \boldsymbol{s}_n^2)$, simplifying the highly nonlinear integrals in (5.5) to

$$\begin{aligned}
\log p_\theta(\boldsymbol{y}_n | \boldsymbol{x}_n) &= \int p(\boldsymbol{y}_n | \boldsymbol{x}_n, \boldsymbol{w}_n) p(\boldsymbol{w}_n) \, \mathrm{d}\boldsymbol{w}_n \\
&\approx \int p(\boldsymbol{y}_n | \boldsymbol{f}_n^L) \mathcal{N}(\boldsymbol{f}_n^L | \boldsymbol{m}_n, \boldsymbol{s}_n^2) \, \mathrm{d}\boldsymbol{f}_n^L \\
&= \mathcal{N}\Big(\boldsymbol{y}_n | \boldsymbol{m}_n, \frac{1}{\beta} + \boldsymbol{s}_n^2\Big),
\end{aligned}$$

where $\boldsymbol{m}_n, \boldsymbol{s}_n^2$ are the mean and variance of $\boldsymbol{f}_n^L$, and $\beta$ is the precision of the normal regression likelihood.

EXTENSIONS. Similar to our discussion on the VBP approach in Chapter 3, adaptations of CLT-based recursive moment matching to many other activation types and skip connections are feasible without further approximations. Max pooling can also be incorporated using approximations but have also been shown to be replaceable altogether by strided convolutions without a performance loss (Springenberg et al., 2015). Deeper networks tend to require normalization procedures, which are not directly amenable to this moment matching. However, one can also compute tractable moments for activation functions such as the ELU (Clevert et al., 2016), which has not been done to our knowledge. We show how to compute the required moments at the end of this chapter.

## 5.2 BAYESIAN EVIDENTIAL DEEP LEARNING (BEDL)

We will now introduce the concept of evidential deep learning (Sensoy et al., 2018) and how to combine it with the marginal likelihood approach.

### 5.2.1 *Evidential Deep Learning*

We can see classification with the cross-entropy loss in the standard deep learning setup as performing maximum-likelihood learning with a categorical likelihood parameterized by a neural net. *Evidential Deep Learning (EDL)* (Sensoy et al., 2018) generalize this setup by instead parameterizing a prior to this categorical likelihood by a neural net $f(\cdot; \boldsymbol{w})$ with deterministic weights $\boldsymbol{w}$, which we can see formally as

$$\begin{aligned} \boldsymbol{\lambda}_n | \boldsymbol{x}_n &\sim p(\boldsymbol{\lambda}_n | \boldsymbol{x}_n), \\ \boldsymbol{y}_n | \boldsymbol{\lambda}_n &\sim p(\boldsymbol{y}_n | \boldsymbol{\lambda}_n), \ \ \forall n. \end{aligned}$$

This way, the model explicitly accounts for the *distributional uncertainty* which may arise due to a mismatch between the train and test data distributions (Quionero-Candela et al., 2009). For classification, a natural choice for the prior $p(\boldsymbol{\lambda}_n | \boldsymbol{w}, \boldsymbol{x}_n)$ on the categorical likelihood is a Dirichlet distribution, and the corresponding model becomes

$$\begin{aligned} \boldsymbol{\lambda}_n | \boldsymbol{x}_n &\sim \mathcal{D}ir\big(\boldsymbol{\lambda}_n | \boldsymbol{\alpha}_n(\boldsymbol{x}_n)\big), \\ \boldsymbol{y}_n | \boldsymbol{\lambda}_n &\sim \mathcal{C}at\big(\boldsymbol{y}_n | \boldsymbol{\lambda}_n\big), \qquad \forall n, \end{aligned}$$

where Sensoy et al. (2018) use $\boldsymbol{\alpha}_n(\boldsymbol{x}_n) \triangleq \max(0, f(\boldsymbol{x}_n; \boldsymbol{w})) + 1$ for some arbitrary neural net architecture $f(\cdot; \cdot)$ to ensure that the positivity constraint on the paramters of the Dirichlet distribution is fulfilled.[2]

To train an EDL model, Sensoy et al. (2018) use a one-hot encoding of the target class and minimize the expected sum of squares between $\boldsymbol{y}_n$ and $\boldsymbol{\lambda}_n$ (known as the Brier score) with an additional regularizing Kullback-Leibler (KL) divergence on the $\boldsymbol{\lambda}_n$

$$\begin{aligned} \mathcal{L}_{\text{EDL}} = \sum_{n=1}^{N} & \mathbb{E}_{p(\boldsymbol{\lambda}_n | \boldsymbol{x}_n)} \left[ ||\boldsymbol{y}_n - \boldsymbol{\lambda}_n||^2 \right] \\ & + \text{KL}\left( \mathcal{D}ir(\boldsymbol{\lambda}_n | \tilde{\boldsymbol{\alpha}}_n) \parallel \mathcal{D}ir(\boldsymbol{\lambda}_n | (1, \dots, 1)) \right), \end{aligned}$$

with the modification of using $\tilde{\boldsymbol{\alpha}}_n = \boldsymbol{y}_n + (1 - \boldsymbol{y}_n) \circ \boldsymbol{\alpha}_n$. This removes the regularization induced by the KL term from the $\alpha$ belonging to the true class in the $n$-th instance.

This loss is, up to constant scaling factors, very similar to a variational inference approach maximizing the evidence lower bound. Assuming a normal likelihood $p(\boldsymbol{y}_n | \boldsymbol{\lambda}_n)$ with precision $\beta$, a prior $p(\boldsymbol{\lambda}_n) = \mathcal{D}ir\big(\boldsymbol{\lambda}_n | (1, \dots, 1)\big)$, and a variational posterior $q(\boldsymbol{\lambda}_n) = \mathcal{D}ir\big(\boldsymbol{\lambda}_n | \boldsymbol{\alpha}_n\big)$, we have the ELBO

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{w}) = -\sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\lambda}_n)} \left[ \log p(\boldsymbol{y}_n | \boldsymbol{\lambda}_n) \right] + \text{KL}\left( q(\boldsymbol{\lambda}_n) \parallel p(\boldsymbol{\lambda}_n) \right)$$

2 We will instead rely on the transformation $\boldsymbol{\alpha}_n(\boldsymbol{x}_n) \triangleq \exp(f(\boldsymbol{x}_n; \boldsymbol{w}))$ throughout the experiments to be closer to the usual interpretation of the neural net outputs being in the log-space followed by a transformation in the probability simplex via the softmax function.

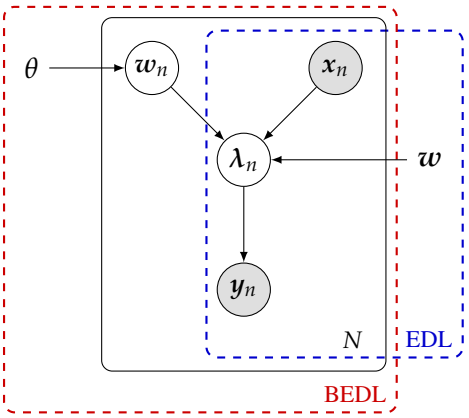



**Figure 5.2:** The generative model as (implicitly) by EDL and our proposed BEDL.

$$\stackrel{c}{=} \sum_{n=1}^{N} \frac{\beta}{2} \mathbb{E}_{q(\lambda_n)} \left[ ||y_n - \lambda_n||_2^2 \right] + \mathrm{KL}\left( q(\lambda_n) \parallel p(\lambda_n) \right), \quad (5.6)$$

where both the expectation as well as the KL terms are analytically tractable for the chosen distributions. The weights of the net so far have been treated as deterministic parameters. In the following subsections, we will discuss how to adapt this setup to probabilistic weights by using the BLNN structure introduced before.

### 5.2.2 *Bayesian Evidential Deep Learning*

We adapt this framework by assigning a local prior on the EDL weights $p_\theta(w_n)$, where we in the experiments will assume the weights to be normal distributed such that we have $w_n \sim p_\theta(w_n) = \mathcal{N}\left(w_n | \mu, \mathrm{diag}(\sigma^2)\right)$, i.e. $\theta = (\mu, \sigma^2)$. The generative model is then

$$\begin{aligned} w_n &\sim \mathcal{N}\left(w_n | \mu, \mathrm{diag}(\sigma^2)\right), \\ \lambda_n | w_n, x_n &\sim \mathcal{D}ir\left(\lambda_n | \alpha_n(x_n; w_n)\right), \\ y_n | \lambda_n &\sim \mathcal{C}at(y_n | \lambda_n), \qquad \forall n. \end{aligned}$$

As a second modification, instead of a mean-square-error objective, we optimize the marginal likelihood, which amounts to employing a BLNN as a prior on the likelihood and marginalizing over all $w_n$ as well as $\lambda_n$. We name the eventual model that combines BLNN with EDL *Bayesian Evidential Deep Learning (BEDL)*.

By virtue of the localized weights, the marginal likelihood of BEDL factorizes across data points. This allows us to consider additive data point specific marginal log-likelihoods, which is the central source of its scalability,

$$\begin{aligned} &\log p_\theta(y|X) \\ &\quad = \log \int \prod_n p(y_n|\lambda_n) p(\lambda_n | w_n, x_n) p_\theta(w_n) \, \mathrm{d}\lambda_1 \ldots \mathrm{d}\lambda_N \, \mathrm{d}w_1 \ldots \mathrm{d}w_N \end{aligned}$$

$$
\begin{aligned}
&= \sum_n \log \int p(\boldsymbol{y}_n|\boldsymbol{\lambda}_n) p(\boldsymbol{\lambda}_n|\boldsymbol{w}_n, \boldsymbol{x}_n) p_\theta(\boldsymbol{w}_n)\, \mathrm{d}\boldsymbol{\lambda}_n\, \mathrm{d}\boldsymbol{w}_n \\
&\approx \sum_n \log p_\theta(\boldsymbol{y}_n|\boldsymbol{x}_n). \qquad (5.7)
\end{aligned}
$$

The marginalization of $\boldsymbol{\lambda}_n$ on the last step can be performed analytically under conjugacy or can be efficiently approximated by Taylor expansion or Monte Carlo sampling, while one can employ the moment matching approach we discussed to marginalize the weights $\boldsymbol{w}_n$.

For the $C$-class classification task with one-hot encoded categorically distributed targets $\boldsymbol{y}_n \in \{0,1\}^C$ and Dirichlet distributed $\boldsymbol{\lambda}_n$ as in the model above, this gives us for the $n$-th term in (5.7),

$$
\begin{aligned}
&\log \int \left( \int p(\boldsymbol{y}_n|\boldsymbol{\lambda}_n) p(\boldsymbol{\lambda}_n|\boldsymbol{\alpha}_n)\, \mathrm{d}\boldsymbol{\lambda}_n \right) \mathcal{N}(\boldsymbol{f}_n^L|\boldsymbol{m}_n, \boldsymbol{s}_n^2)\, \mathrm{d}\boldsymbol{f}_n^L \\
&\qquad = \log \mathbb{E}_{\mathcal{N}(\boldsymbol{f}_n^L|\boldsymbol{m}_n, \boldsymbol{s}_n^2)} \left[ \prod_{c=1}^{C} \left( \frac{\alpha_{nc}}{\alpha_{n0}} \right)^{y_{nc}} \right], \qquad (5.8)
\end{aligned}
$$

where we use the parametrization $\boldsymbol{\alpha}_n = (\alpha_{n1}, ..., \alpha_{nC}) = \exp(\boldsymbol{f}_n^L)$, and $\alpha_{n0} = \sum_c \alpha_{nc}$. The main computational bottleneck consists of the marginalization of the weights. Circumenting this by the analytical CLT-based moment matching through the neural net layers, the final expectation is cheap enough to be efficiently approximated by sampling.

UNCERTAINTY DECOMPOSITION. Following Depeweg et al. (2018) one can use the law of total variance to decompose the predictive variance of the final marginal for a data pair $(\boldsymbol{x}, \boldsymbol{y})$ as follows

$$
\mathrm{var}\left[\boldsymbol{y}|\boldsymbol{x}\right] = \mathrm{var}_{\boldsymbol{w}}\left[\mathbb{E}\left[\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{x}\right]\right] + \mathbb{E}_{\boldsymbol{w}}\left[\mathrm{var}\left[\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{x}\right]\right].
$$

Here the first term, $\mathrm{var}_{\boldsymbol{w}}\left[\mathbb{E}\left[\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{x}\right]\right]$, focuses on the contribution to this predictive uncertainty by the variance over the network weights, i.e. the epistemic uncertainty. In contrast, the second, $\mathbb{E}_{\boldsymbol{w}}\left[\mathrm{var}\left[\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{x}\right]\right]$, represents the remaining variance in the likelihood for the average weights. After having marginalized over $\boldsymbol{\lambda}$, the mean and variance $\mathbb{E}\left[\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{x}\right]$, $\mathrm{var}\left[\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{x}\right]$, are analytically tractable and are given as

$$
\mathbb{E}\left[y_c|\boldsymbol{w}, \boldsymbol{x}\right] = \frac{\alpha_c}{\alpha_0} \;\text{ and }\; \mathrm{var}\left[y_c|\boldsymbol{w}, \boldsymbol{x}\right] = \frac{\alpha_c}{\alpha_0}\left(1 - \frac{\alpha_c}{\alpha_0}\right).
$$

The EDL formulation allows for an analytic computation of the predictive variance, however as it considers only deterministic weights, it gets for a set of learned learned parameters $\hat{\boldsymbol{w}}$ the predictive variance

$$
\mathrm{var}\left[\boldsymbol{y}|\boldsymbol{x}\right] = \mathrm{var}\left[\boldsymbol{y}|\hat{\boldsymbol{w}}, \boldsymbol{x}\right],
$$

i.e. only a measure of the aleatoric uncertainty, lacking the epistemic. Our extension allows for the decomposition of the predictive uncertainty maintaining analytical tractability of the approximation to a great extent

$$
\mathrm{var}\left[\boldsymbol{y}|\boldsymbol{x}\right] = \mathrm{var}_{\boldsymbol{w}}\left[\mathbb{E}\left[\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{x}\right]\right] + \mathbb{E}_{\boldsymbol{w}}\left[\mathrm{var}\left[\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{x}\right]\right]
$$

$$\approx \mathrm{var}_{f^L}\left[\mathbb{E}\left[\boldsymbol{y}|\boldsymbol{f}^L,\boldsymbol{x}\right]\right] + \mathbb{E}_{f^L}\left[\mathrm{var}\left[\boldsymbol{y}|\boldsymbol{f}^L,\boldsymbol{x}\right]\right],$$

where the final variance and expectation can be efficiently approximated with samples as discussed above.

## 5.3 A VACUOUS PAC-BAYES BOUND TO REGULARIZE BEDL

Training the objective in (5.7) is effective for fitting a predictor to the observed data. It also naturally provides a learned loss attenuation mechanism. However, as we already hinted at, it lacks a key advantage of the Bayesian modelling paradigm. As the hyperparameters of the weight priors are employed for model fitting, they no longer contribute to training as complexity penalizers. It is well-known from the GP literature that marginal likelihood-based training is prone to overfitting for models with a large number of hyperparameters (Bauer et al., 2016). We address this shortcoming by complementing the marginal likelihood objective of (5.7) with a penalty term derived from learning-theoretic first principles. We tailor the eventual loss only for robust model training and keep it maximally generic across learning setups. This comes at the expense of arriving at a generalization bound that makes a theoretically trivial statement, as it is too loose to give a good generalization guarantee, yet brings significant improvements to training quality, as illustrated in our experiments.

PAC-Bayes bounds have been commonly used for likelihood-free and loss-driven learning settings. A rare exception by Germain et al. (2016) proves the theoretical equivalence of a particular sort of PAC-Bayes bound to variational inference. Similarly to their approach, we keep the notion of a likelihood in our risk definition, but differ in that we adapt our bound to the marginal likelihood.

*True & Empirical risk*

Given a predictor $h$ chosen from a hypothesis class $\mathcal{H}$ as a mapping from $\boldsymbol{x}$ to $\boldsymbol{y}$, we can define the true ($R(h)$) and the empirical ($R_{\mathcal{D}}(h)$) risks as the expected negative marginal likelihood

$$R(h) = -\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\Delta}\left[p\big(\boldsymbol{y}|h(\boldsymbol{x})\big)\right],$$

and its observed counterpart,

$$R_{\mathcal{D}}(h) = -\frac{1}{N}\sum_{n=1}^{N} p\big(\boldsymbol{y}_n|h(\boldsymbol{x}_n)\big),$$

for an observed data set $\mathcal{D}$ of size $N$ which is in turn drawn from an arbitrary and unknown data distribution $\Delta$.

The risks $R(h)$ and $R_{\mathcal{D}}(h)$ are bounded below by $-\max p(y|h(x))$ and above by zero. Although this setting relaxes the common assumption that bounds risk to the $[0,1]$ interval, it is still substantially simpler than the one suggested by Germain et al. (2016), who define

$$R(h) = \mathbb{E}_{x,y\sim\Delta}\left[\log p(y|h(x))\right] \in (-\infty,+\infty).$$

This unboundedness brings severe technical complications, which are no longer relevant to our approach.

Translating Theorem 2.1 by Germain et al. (2009) to our notation gives us that for any data distribution $\Delta$, any set of classifiers $\mathcal{H}$, any $\delta \in (0,1]$, and any convex function $d(\cdot,\cdot)$ we have the following PAC-Bayes bound

$$\mathbb{P}\left\{ d\Big(\mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right], \mathbb{E}_{h\sim Q}\left[R(h)\right]\Big) \leq \frac{\mathrm{KL}\left(Q \parallel P\right) + \log(B/\delta)}{N} \right\} \geq 1-\delta, \tag{5.9}$$

with $B = \mathbb{E}_{\mathcal{D}\sim\Delta}\left[\mathbb{E}_{h\sim P}\left[\exp\left(N d(R_{\mathcal{D}}(h), R(h))\right)\right]\right]$. $Q$ denotes a learnable *posterior* distribution over the space of hypothesis $\mathcal{H}$ and $P$ specifies a *prior* distribution over the same space.

However, in its current form, this bound is neither tractable as a generalization bound on the true risk, neither as a regularizing objective which we aim to utilize for training. For that, we need to choose a suitable $d(\cdot,\cdot)$ and derive a tractable approximation to $B$. $P$ and $Q$ are, in our case, already specified via the generative model we created.

DERIVATION OF $d(\cdot,\cdot)$. The PAC-Bayes framework necessitates a convex and non-negative distance measure for risk evaluations. A common practice is to rescale the risk into the unit interval, define the KL divergence between two Bernoulli distributions as the distance measure, and upper bound its intractable inverse (Germain et al., 2016) using Pinsker's inequality (Tolstikhin and Seldin, 2013; Dziugaite and Roy, 2017). We follow an alternative path. As our risk is bounded but not restricted to the unit interval, we choose our distance measure as $d(r,r') = (r-r')^2$ and avoid Pinsker's inequality step.

Instead, we adapt the KL inversion trick (Seeger, 2002; Reeb et al., 2018) to the Euclidean distance, by defining $d^{-1}(x,\varepsilon) \triangleq \max\{x' : (x-x')^2 = \varepsilon\} = x + \sqrt{\varepsilon}$ for some $\varepsilon \geq 0$. Applying this function to both sides of the inequality in (5.9) with $\mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right]$ as its first argument gives

$$\begin{aligned} & d^{-1}\Big(\mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right], d\big(\mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right], \mathbb{E}_{h\sim Q}\left[R(h)\right]\big)\Big) \\ \leq\ & d^{-1}\Big(\mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right], \tfrac{1}{N}\big(\mathrm{KL}\left(Q \parallel P\right) + \log(B/\delta)\big)\Big), \end{aligned}$$

since $d(\cdot,\cdot)$ and $\mathrm{KL}\left(Q \parallel P\right)$ are each non-negative, and due to $\delta \in (0,1]$ and $\exp(d(\cdot,\cdot)) \geq 0$ we have that $\log(\frac{B}{\delta}) = -\log\delta + \log B \geq 0$, i.e. the non-negativity constraint on the second argument in $d^{-1}(\cdot,\cdot)$ is fulfilled. It then gives us for the left-hand side an upper bound on the true risk

$$\mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right] + \sqrt{d\Big(\mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right], \mathbb{E}_{h\sim Q}\left[R(h)\right]\Big)} \geq \mathbb{E}_{h\sim Q}\left[R(h)\right].$$

In combination with the application of $d^{-1}(\cdot,\cdot)$ to the right-hand side, we can upper bound the true risk as

$$\mathbb{P}\Bigg\{\forall Q \;\; \mathbb{E}_{h\sim Q}\left[R(h)\right] \leq \mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right] + \sqrt{\frac{KL(Q||P)+\log(B/\delta)}{N}}\Bigg\} \geq 1-\delta.$$

This outcome has a similar structure to what an application of Pinsker's inequality would yield to a setup where the risk is restricted to the unit interval, but without such a restriction. To arrive at a tractable bound, we have to further approximate each of the two terms on the right-hand side of the bound.

*Bound on the empirical risk*

For the first term, we have that

$$\begin{aligned}\mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right] &\triangleq -\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{h\sim Q}\left[p\big(\boldsymbol{y}_n|h(\boldsymbol{x}_n)\big)\right] \\ &\leq -\frac{1}{N}\sum_{n=1}^{N}\log\mathbb{E}_{h\sim Q}\left[p\big(\boldsymbol{y}_n|h(\boldsymbol{x}_n)\big)\right] \\ &\approx -\frac{1}{N}\sum_{n=1}^{N}\log p(\boldsymbol{y}_n|\boldsymbol{x}_n),\end{aligned}$$

where the inequality uses that $-\log(u) > -u \;\forall u \in (0,\infty)$ and the final approximation follows via the CLT based marginalization technique.

To get a tractable second term, we need to evaluate $B$ further. For this, we exploit the fact that for any $p(\cdot)$, and $f(\cdot)$, we have the direct relation

$$\mathbb{E}_{p(x)}\left[f(x)\right] \leq \max f(x),$$

which allows us to drop the expectations and get for our chosen $d(\cdot,\cdot)$

$$B \triangleq \mathbb{E}_{\mathcal{D}\sim\Delta}\left[\mathbb{E}_{h\sim P}\left[e^{N(R_{\mathcal{D}}(h)-R(h))^2}\right]\right] \leq \max e^{N(R_{\mathcal{D}}(h)-R(h))^2}.$$

*Bound on B for classification*

For a multiclass classification task, the likelihood is bounded into the interval $[0,1]$, and such the true and empirical risks are as well such that we have

$$\max\left(R_{\mathcal{D}}(h)-R(h)\right)^2 = (0-1)^2 = 1,$$

which gives a final upper bound on $B$ as

$$B \leq \exp(N) \quad \Rightarrow \quad \log B \leq N.$$

*Bound on B for regression*

For a regression tasks with a normal homoscedastic likelihood $\mathcal{N}(y|h(x),\beta^{-1})$ we have similarly that the risks can be upper bounded by 0 and lower-bounded by $-\mathcal{N}(y=\mu|\mu,\beta^{-1})$, that is by the density at the mode of a normal distribution with a precision of $\beta$. Hence,

$$B \leq \exp\left(N\big(0-\mathcal{N}(\mu,\beta^{-1})\big)^2\right) \quad \Rightarrow \quad \log B \leq N\frac{\beta}{2\pi}.$$

Combining these results gives us the final objective used to train BEDL as

$$-\frac{1}{N}\sum_{n=1}^{N}\log p_\theta(\boldsymbol{y}_n|\boldsymbol{x}_n)+\sqrt{\frac{\mathrm{KL}\left(Q\parallel P\right)-\log\delta}{N}+\frac{\log\max(B)}{N}}, \quad (5.10)$$

which contains the marginal likelihood objective discussed above as its first term and a regularizer in the second. This additional term resembles the KL term in the EDL loss, but gives via the PAC-Bayes approach a theoretically-grounded mechanism to incorporate regularization.

*The KL term for classification*

For a $C$-class classification we follow EDL, and use in the regularization term $P=\mathcal{D}ir\big(\boldsymbol{\lambda}|(1,\ldots,1)\big)$, i.e. the assumption that each class is equally likely, as the regularizing distribution. Given $Q=\mathcal{D}ir(\boldsymbol{\lambda}|\boldsymbol{\alpha})$, the Kullback-Leibler divergence is analytically tractable and given as

$$\begin{aligned}&\mathrm{KL}\left(\mathcal{D}ir(\boldsymbol{\lambda}|\boldsymbol{\alpha})\parallel\mathcal{D}ir(\boldsymbol{\lambda}|(1,\ldots,1))\right)=\\&\qquad\log\left(\frac{\Gamma(\sum_c\alpha_c)}{\Gamma(C)\prod_c\Gamma(\alpha_c)}\right)+\sum_{c=1}^{C}(\alpha_c-1)\Big(\psi(\alpha_c)-\psi\big(\sum_c\alpha_c\big)\Big),\end{aligned}$$

with the digamma function $\psi(x)\triangleq\frac{d}{da}\log\Gamma(x)$.

LOOSENESS OF THE BOUND. It should finally be noted that while we use the PAC-Bayes theory to derive and motivate the final objective, it should no longer be used in original PAC-Bayes motivation, that is to get some generalization guarantees. The sequence of approximation steps we require result in a loose bound that is trivially fulfilled. As such it no longer serves as a useful indicator of the generalization performance. Rather its justification lies primarily in its regularizing function.

### 5.3.1 *Hyperprior*

We close this section with a short discussion on one obvious direct objection to the approach we discussed. The PAC-Bayes based model seems nice and principled, but why would one not just make use of the hierarchical structure of a Bayesian generative model and go one level higher? That is, introduce some additional hyperpriors, reintroducing the missing regularizing component of the marginal likelihood objective that way.

Such a hierarchical model would have the following structure

$$\begin{aligned}\theta&\sim p(\theta),\\ \boldsymbol{w}|\theta&\sim\prod_n p_\theta(\boldsymbol{w}_n),\\ \boldsymbol{\lambda}|\boldsymbol{w},\boldsymbol{X}&\sim\prod_n p(\boldsymbol{\lambda}_n|\boldsymbol{w}_n,\boldsymbol{x}_n),\\ \boldsymbol{y}|\boldsymbol{\lambda}&\sim\prod_n p(\boldsymbol{y}_n|\boldsymbol{\lambda}_n),\qquad\forall n.\end{aligned}$$

The marginal likelihood to be optimized over is then given as

$$\log p(\boldsymbol{y}, \theta | \boldsymbol{X}) = \sum_n \log \int p(\boldsymbol{y}_n | \boldsymbol{\lambda}_n) p(\boldsymbol{\lambda}_n | \boldsymbol{w}_n, \boldsymbol{x}_n) p(\boldsymbol{w}_n | \theta) \, \mathrm{d}\boldsymbol{\lambda}_n \, \mathrm{d}\boldsymbol{w}_n + \log p(\theta).$$

In this objective the first term can be approximated and evaluated as before, returning our by now familiar marginal-likelihood formulation. However, the second term $\log p(\theta)$ provides us with some hopefully regularizing structure. The form of this hyperprior will vary depending on the problem at hand, but consider for example our normally distributed weights, where we assume the $i$-th weight of the BNN to follow

$$w_n^i | \theta_i \sim \mathcal{N}(w_n^i | \mu_i, \sigma_i^2), \quad \text{where } \theta_i = (\mu_i, \sigma_i^2).$$

A choice for the prior $p(\theta_i)$ is then to assume

$$\begin{aligned} p(\theta_i) &= p(\mu_i) p(\sigma_i^2) \\ &= \mathcal{N}(\mu | 0, \alpha_0^{-1}) \mathcal{I}nv\mathcal{G}am(\sigma^2 | a_0, b_0). \end{aligned}$$

The choice of the hyperparameters $\alpha_0$, $a_0$, and $b_0$ then control the strength of the regularization. We explore this variant of the proposed model in the regression experiments, but the results there show it to perform worse than the PAC-Bayes based objective.

## 5.4 GENERALIZATION TO REGRESSION

The EDL formulation was introduced by Sensoy et al. (2018) only for the task of classification, which we focused on in our discussion so far. However, we can also extend the main motivation of the approach to the task of regression.

We place a normal likelihood over the targets, treating the $\lambda_n$ as the mean and keep a fixed precision $\beta$, as necessitated by our PAC-Bayes derivations. Placing another normal distribution over the $\lambda_n$ whose mean and variance are determined by a BLNN gives us as the generative model

$$\begin{aligned} \boldsymbol{w}_n &\sim p_\theta(\boldsymbol{w}_n), \\ \lambda_n | \boldsymbol{w}_n, \boldsymbol{x}_n &\sim \mathcal{N}\Big(\lambda_n \Big| f_1(\boldsymbol{x}_n; \boldsymbol{w}_n), \exp\left(f_2(\boldsymbol{x}_n; \boldsymbol{w}_n)\right)\Big), \\ y_n | \lambda_n &\sim \mathcal{N}\left(y_n | \lambda_n, \beta^{-1}\right), \qquad \forall n. \end{aligned}$$

For the $n$-th sample in (5.7) after performing the moment matching until the last layer we have the marginal for $\lambda_n$ given as

$$p(\lambda_n) \approx \int \mathcal{N}\big(\lambda_n | f_{n1}^L, \exp(f_{n2}^L)\big) \mathcal{N}(\boldsymbol{f}_n^L | \boldsymbol{m}_n, \boldsymbol{s}_n^2) \, \mathrm{d}\boldsymbol{f}_n^L,$$

with $\boldsymbol{f}_n^L = (f_{n1}^L, f_{n2}^L)$, $\boldsymbol{m}_n = (m_{n1}, m_{n2})$, and $\boldsymbol{s}_n^2 = (s_{n1}^2, s_{n2}^2)$ (the moment matching mean and variance). This integral can then be further approximated with a final moment matching step.

Dropping the indices $n$ and $L$ for notational simplicity, we get

$$\begin{aligned}
\mathbb{E}_{p(\lambda)}[\lambda] &= \int \lambda p(\lambda)\,\mathrm{d}\lambda \\
&\approx \iint \lambda \mathcal{N}(\lambda|f_1, \exp(f_2))\mathcal{N}(\boldsymbol{f}|\boldsymbol{m}, \boldsymbol{s}^2)\,\mathrm{d}\boldsymbol{f}\,\mathrm{d}\lambda \\
&= \int \left(\int \lambda \mathcal{N}(\lambda|f_1, \exp(f_2))\,\mathrm{d}\lambda\right) \mathcal{N}(\boldsymbol{f}|\boldsymbol{m}, \boldsymbol{s}^2)\,\mathrm{d}\boldsymbol{f} \\
&= \int f_1 \mathcal{N}(\boldsymbol{f}|\boldsymbol{m}, \boldsymbol{s}^2)\,\mathrm{d}\boldsymbol{f} = m_1.
\end{aligned}$$

For the variance term we rely on the law of total variance and have

$$\begin{aligned}
\mathrm{var}_{p(\lambda)}[\lambda] &= \mathbb{E}_{p(\boldsymbol{f})}\left[\mathrm{var}_{p(\lambda|\boldsymbol{f})}[\lambda]\right] + \mathrm{var}_{p(\boldsymbol{f})}\left[\mathbb{E}_{p(\lambda|\boldsymbol{f})}[\lambda]\right] \\
&\approx \mathbb{E}_{p(\boldsymbol{f})}[\exp(f_2)] + \mathrm{var}_{p(\boldsymbol{f})}[f_1] \\
&= \int \exp(f_2)\mathcal{N}(f_2|m_2, \sigma_2^2)\,\mathrm{d}f_2 + s_1^2 \\
&= \exp(m_2 + s_2^2/2) + s_1^2,
\end{aligned}$$

where the last integral is given as the expectation of a log-normal random variable. Altogether we end up with the desired normal approximation

$$p(\lambda_n) \approx \mathcal{N}\Big(\lambda_n \big| m_{n1}, s_{n1}^2 + \exp(m_{n2} + s_{n2}^2/2)\Big).$$

This then allows us to compute the log marginal likelihood as

$$\begin{aligned}
\log p(y_n|\boldsymbol{x}_n) &\approx \log \int \mathcal{N}(y_n|\lambda_n, \beta^{-1}) \\
&\qquad \cdot \left(\int \mathcal{N}(\lambda_n|f_{n1}^L, \exp(f_{n2}^L))\mathcal{N}(\boldsymbol{f}_n^L|\boldsymbol{m}_n, \boldsymbol{s}_n^2)\,\mathrm{d}\boldsymbol{f}_n^L\right)\mathrm{d}\lambda_n \\
&\approx \log \int \mathcal{N}(y_n|\lambda_n, \beta^{-1})\mathcal{N}\Big(\lambda_n \big| m_{n1}, s_{n1}^2 + \exp(m_{n2} + s_{n2}^2/2)\Big)\,\mathrm{d}\lambda_n \\
&= \log \mathcal{N}\Big(y_n \big| m_{n1}, \beta^{-1} + s_{n1}^2 + \exp(m_{n2} + s_{n2}^2/2)\Big),
\end{aligned}$$

where the last equality follows directly from standard results on normal distributions.

Contrary to the classification case where the final step requires samples, we stay completely sampling-free for regression. As discussed above, we bound $\max B$ in (5.10) by exploiting that $\beta$ is fixed prior to training and obtain $\frac{\log \max B}{N} \leq \frac{\beta}{2\pi}$ as a bound, which, as for the classification case, gives only a trivial performance guarantee (exceeding the maximum possible risk) but provides a justified training scheme.

## 5.5 RELATED WORK

CLT-BASED MOMENT MATCHING OF LOCAL WEIGHT REALIZATIONS. The objective for variational inference on BNNs (Kingma et al., 2015; Gal

and Ghahramani, 2016b; Wu et al., 2019), optimizing a global variational posterior $q(\boldsymbol{w})$, consists of a computationally intractable $\mathbb{E}_{q(\boldsymbol{w})}\left[\log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w})\right]$ that decomposes across data points. Fast dropout (Wang and Manning, 2013) approximates these terms via local reparameterization with moment matching. The same local reparameterization has been later combined with a KL term to perform mean-field variational inference via Monte Carlo sampling (Kingma et al., 2015) or moment matching (Wu et al., 2019). Our Bayesian local neural network formulation is akin to an amortized variational inference approach learning a single global set of posterior parameters $\phi$ for the variational posterior approximation $q_\phi(\boldsymbol{w})$. We use the same trick to marginalize the local weights, which keeps the machinery similar until the top-most step where the $\log(\cdot)$ and $\mathbb{E}\left[\cdot\right]$ operations is swapped. This small change, however, has a large impact on the quality of uncertainty estimations.

WIDE NEURAL NETS AS GAUSSIAN PROCESSES. A Gaussian Process's equivalence to a weight-marginalized BNN with a single infinitely wide hidden layer has been discovered long ago (Neal, 1995) using the multivariate version of CLT. This result has later been generalized to multiple dense layers (G. Matthews et al., 2018; Lee et al., 2018a), as well as to convolutional layers (Garriga-Alonso et al., 2019). The asymptotic treatment of the neuron count makes this approach exact at the expense of a lack of neuron-specific parameterization. The eventual GP has few hyperparameters to train—however, a prohibitively expensive covariance matrix to calculate. We employ the same training method on a middle ground where the number of hyperparameters is twice the number of parameters in a deterministic net of the same size. The cross-covariances across data points are not explicitly modelled.

PREDICTIVE DENSITY MODELLING. The recently introduced family of models known as Neural Processes (Garnelo et al., 2018a,b) follow a GP inspired approach of learning the predictive density using neural networks as part of the mapping from input to output space, relying on the incorporation of the input data as context points for the predictive distribution of a test point. Earlier work on prior networks (Malinin and Gales, 2018) parameterizes a prior to a classification-specific likelihood with deterministic neural nets, hence, discards model uncertainty. Additionally, they require samples from another domain to learn distributional awareness. BEDL reformulates prior networks independently from the output structure, extends them to support also model uncertainty, and introduces a principled scheme for their training.

## 5.6 EXPERIMENTS

We evaluate the proposed model and its PAC-Bayes-regularized version on a diverse selection of regression and classification tasks. Additional information on experimental details not discussed here is provided in the further details section at the end of this chapter.[3]

3 We provide an implementation of the model under `github.com/manuelhaussmann/bedl`.

### 5.6.1 *Regression*

As in the case of VBP, we evaluate the performance on the regression task on the standard experimental protocol introduced by Hernández-Lobato and Adams (2015). The tasks consist of fitting a net with a single hidden layer with 50 units and ReLU nonlinearities[4] to eight different UCI data sets of varying difficulty and size.

We evaluate three versions of the proposed model.

*BEDL*

i) *BEDL* refers to the basic model, that performs the CLT based marginalization and optimizes the resulting marginal likelihood

$$\max_{\phi} \sum_{n=1}^{N} \log p_{\phi}(y_n|\boldsymbol{x}_n);$$

*BEDL-Hyper*

ii) *BEDL-Hyper* gives the baseline of encorporating a hyperprior-based regularization

$$\max_{\phi} \sum_{n=1}^{N} \log p_{\phi}(\boldsymbol{y}_n|\boldsymbol{x}_n) + \log p(\phi);$$

*BEDL-PAC*

iii) *BEDL-PAC* finally represents the proposed model with the PAC-based objective

$$\min_{\phi} -\frac{1}{N} \sum_{n=1}^{N} \log p_{\phi}(y_n|\boldsymbol{x}_n) + \sqrt{\frac{\mathrm{KL}\left(Q \parallel P\right) - \log \delta + \log \max B}{N}}.$$

The hypothesis class in this case is over the regularization parameters $P \triangleq p(\boldsymbol{\lambda}) = \prod_n \mathcal{N}(\lambda_n|0, \alpha^{-1})$, with a fixed prior precision $\alpha$, while $Q$ is given as $Q \triangleq p(\boldsymbol{\lambda}) = \prod_n \int p(\lambda_n|\boldsymbol{f}_n)p(\boldsymbol{f}_n)d\boldsymbol{f}_n$. This gives us an analytically tractable KL divergence between two normal distributions.

All three variants are sampling-free in the regression case. We compare them against two other sampling-free variants for training BNNs that are also CLT based, representing the state-of-the-art in such approaches. *Probabilistic Back-Propagation* (PBP) (Hernández-Lobato and Adams, 2015) and *Deterministic Variational Inference* (DVI) (Wu et al., 2019) both utilize similar CLT-based moment matching techniques, relying on them as part of an expectation propagation and variational inference setup, respectively. For completeness, we also compare against the two most popular sampling-based alternatives, *Variational Dropout* (VarOut) (Kingma et al., 2015; Molchanov et al., 2017) and *MC Dropout* (Gal and Ghahramani, 2016b). We include for comparison the results of fitting a sparse Gaussian Process to the same setup.

We summarize the results of 20 random train-test splits comprising 90% and 10% of the samples, respectively, in Table 5.1. Our three variants and Variational Dropout share the same train/test splits. For the other methods, we report

4 Except on the larger `protein`, for which we use 100 hidden units.

**Table 5.1: Regression.** Average test log-likelihood ± standard error over 20 random train/test splits. N/d give the number of data points in the complete data set and the number of input feature. The sparse GP results are cited from (Bui et al., 2016) and serve as a comparison. VarOut relies on our own implementation. The best performing BNN in each case is marked in bold.

| | boston | concrete | energy | kin8nm | naval | power | protein | wine |
|---|---|---|---|---|---|---|---|---|
| $N/d$ | 506/13 | 1030/8 | 768/8 | 8192/8 | 11934/16 | 9568/4 | 45730/9 | 1599/11 |
| Sparse GP | −2.22±0.07 | −2.85±0.02 | −1.29±0.01 | 1.31±0.01 | 4.86±0.04 | −2.66±0.01 | −2.95±0.05 | −0.67±0.01 |
| MC Dropout | −2.46±0.25 | −3.04±0.09 | −1.99±0.09 | 0.95±0.03 | 3.80±0.05 | −2.89±0.01 | −2.80±0.05 | −0.93±0.06 |
| VarOut | −2.63±0.02 | −3.15±0.02 | −3.29±0.00 | 1.09±0.01 | 5.50±0.03 | −2.82±0.01 | −2.90±0.01 | **−0.88±0.02** |
| PBP | −2.57±0.09 | −3.16±0.02 | −2.04±0.02 | 0.90±0.01 | 3.73±0.01 | −2.84±0.01 | −2.97±0.00 | −0.97±0.01 |
| DVI | **−2.41±0.02** | −3.06±0.01 | −1.01±0.06 | 1.13±0.00 | **6.29±0.04** | −2.80±0.00 | −2.84±0.01 | −0.90±0.01 |
| BEDL | −2.45±0.08 | −3.09±0.06 | −0.87±0.10 | 1.12±0.01 | 5.76±0.07 | −2.80±0.01 | −2.82±0.01 | −0.93±0.01 |
| BEDL-Hyper | −2.57±0.04 | −3.30±0.01 | −2.59±0.02 | 0.44±0.00 | 3.69±0.00 | −2.98±0.01 | −3.00±0.00 | −1.00±0.01 |
| BEDL-PAC | −2.43±0.06 | **−3.02±0.02** | **−0.73±0.04** | **1.15±0.01** | 5.60±0.11 | **−2.79±0.01** | **−2.77±0.01** | −0.90±0.01 |

the average test log-likelihoods reported in the respective papers. The sparse GP results are cited from Bui et al. (2016).

Our proposed *BEDL-PAC* improves upon the other approaches in the majority of the datasets. In particular, it improves upon the plain BEDL in all data sets except for one. On the other hand, the hyperprior-based approach struggles a lot more, deteriorating the performance on all eight data sets. As expected, the sparse GP with 50 inducing points, approximating a BNN with one infinitely wide hidden layer (Neal, 1995), performs very competitively as well.

### 5.6.2 *Classification and Out-of-Distribution Detection*

One common proxy task to evaluate the predictive uncertainty quality of neural nets is to evaluate their out-of-distribution (OOD) sample detection performance. That is, a model trained on one distribution of data should correctly assign a low likelihood to samples from a different domain instead of confidently assigning it to one of the classes from the training distribution.

We evaluate classification and out-of-distribution sample detection performance of *BEDL-PAC* on image classification with deep architectures, adhering to the protocol used in prior work (Louizos and Welling, 2017; Sensoy et al., 2018). To do that, we train LeNet-5 networks on the MNIST train split, evaluate their classification accuracy on the MNIST test split as the in-distribution task, and measure their uncertainty on the Fashion-MNIST[5] data set as the out-of-distribution task. We expect a perfect model to predict true classes with high accuracy on the in-distribution data and always predict a uniform probability

5 Due to the license status of the not-MNIST data which was used in the cited prior work conflicting with the affiliation of a collaborator on the original paper that this chapter is based on, we had to change the setup in this respect, using instead Fashion-MNIST as the closest substitute, keeping the rest of the experimental framework identical.

**Table 5.2: Classification and OOD Detection.** Test error and the area under curve of the empirical CDF (ECDF-AUC) of the predictive entropies on two pairs of datasets. Smaller values are better for both metrics.

| | MNIST (In distribution) | Fashion-MNIST (Out-of-distribution) | CIFAR 1-5 (In distribution) | CIFAR 6-10 (Out-of-distribution) |
|---|---|---|---|---|
| | Test Error (%) | ECDF-AUC | Test Error (%) | ECDF-AUC |
| MC Dropout | 1.12 | 0.429 | **18.36** | 0.946 |
| VarOut | 1.47 | 1.381 | 33.94 | 0.673 |
| DVI | 0.72 | 1.318 | 23.32 | 1.251 |
| EDL | 1.08 | 0.132 | 20.34 | 0.451 |
| BEDL | 0.81 | 1.512 | 24.38 | 1.253 |
| BEDL-PAC | **0.66** | **0.055** | 20.02 | **0.083** |

mass on the out-of-distribution data. To get a scalar performance measure, we use the following evaluation measure. The desired performance on out-of-distribution data is equivalent to maximize the entropy over the prediction. Computing the predictive entropy over all the test data and computing the area under curve of the empirical CDF (ECDF-AUC) of this entropy then gives us a scalar summary that is minimized if the predictive entropy is maximized for each data point.

We perform the same experiment on CIFAR10 using the first five classes for the in-distribution task and treating the rest as out-of-distribution. We use $P \triangleq \mathcal{D}ir(\boldsymbol{\lambda}|(1,\ldots,1))$ as the regularization prior on the class assignment parameters, which has the uniform probability mass on its mean, encouraging an OOD alarm in the absence of contrary evidence. In Table 5.2, we compare BEDL-PAC against EDL (Sensoy et al., 2018), a state of the art approach in neural net-based uncertainty quantification and also the non-Bayesian and heuristically trained counterpart of BEDL-PAC. We consider EDL also as a special case of Prior Networks (Malinin and Gales, 2018) that does not need to rely on OOD data during training time, commensurate for our training assumptions. We evaluate MC Dropout, VarOut, and DVI as baselines in this setup. BEDL-PAC improves state of the art in all four cases except the CIFAR10 in-distribution task, where it ranks second after the prediction time weight sampling-based MC Dropout. Additionally, BEDL-PAC detects the OOD samples with better ECDF-AUC scores than EDL.

### 5.6.3 *Comparison to GP-based Variants*

We evaluate the impact of local weight realization on prediction performance by comparing BEDL-PAC to GPs with kernels derived from BNNs with global weight realizations (Neal, 1995; Lee et al., 2018a; Garriga-Alonso et al., 2019) on MNIST and CIFAR10 data sets. It is technically not possible to perform this evaluation in a fully comparable setup, as these baselines assume infinitely many neurons per layer and do not have weight-specific degrees of freedom. Furthermore, Garriga-Alonso et al. (2019) perform neural architecture search

**Table 5.3: Comparison to GP Variants.** Test error in % on two image classification tasks. BEDL reaches a lower error rate than previously proposed neural net-based GP constructions by two convolutional layers with 96 filters of size $5 \times 5$ and stride 2. BEDL converges in 50 epochs, amounting to circa 30 minutes of training time on a single GPU. The GP alternatives have been reported to have significantly larger runtime and memory requirements. The GP results are cited from Garriga-Alonso et al. (2019)

| | **MNIST** | **CIFAR10** |
|---|---|---|
| NNGP | 1.21 | 44.3 |
| Convolutional GP | 1.17 | 35.4 |
| ConvNet GP | 1.03 | - |
| Residual CNN GP | 0.96 | - |
| ResNet GP | 0.84 | - |
| BEDL (Ours) | 0.91 | 34.20 |
| BEDL-PAC (Ours) | **0.63** | **32.47** |

and Lee et al. (2018a) use only part of the CIFAR10 training set, reporting that the rest does not fit into the memory of a powerful workstation. We nevertheless view the performance scores reported in these papers as practical upper bounds and provide a qualitative comparison. For the choice of neural net depth, we take NNGP (Lee et al., 2018a) as a reference and devise a contrarily thin two-layer convolutional BEDL-PAC network. The results and the architectural details are summarized in Table 5.3. BEDL and BEDL-PAC can reach lower error rates using significantly less computational resources.

### 5.6.4 *Computational Cost*

Table 5.4 summarizes the computational cost of the considered approaches. MC Dropout and VarOut can quantify uncertainty only by taking samples across weights, which increases the prediction cost linearly to the sample count. DVI and BEDL-PAC perform the forward pass during both training and prediction time via analytical moment matching at double and triple costs, respectively. Both methods have sampling costs for intractable likelihoods.[6] BEDL-PAC may also have another additive per-data-point sampling cost for calculating intractable functional mapping regularizers. Favourably, both of these overheads are only additive to the forward pass cost, i.e. sampling time is independent of the neural net depth. Hence they do not set a computational bottleneck. The training and prediction cost of BEDL-PAC is three times that of EDL which builds on deterministic nets. However, it provides substantial improvements in both prediction accuracy and uncertainty quantification.

6 Even this sampling step could be avoided by a suitable Taylor approximation, see for example, the corresponding discussion in the VBP chapter. As the added approximation error was more detrimental to model performance than a cheap MC approach in preliminary experiments, we stay with the latter for both.

**Table 5.4: Computational Cost.** Per data point computational cost analysis in FLOPs. **F:** Forward pass cost of a deterministic neural net. **W:** Number of weights in the net. **L:** Analytical calculation cost for the exact or approximate likelihood or the loss term. **S:** Number of samples taken for approximation. **R:** The cost of the regularization term per unit (weight or data point).

| | **Training per iteration** | **Prediction** |
|---|---|---|
| MC Dropout | $\mathcal{O}\big((F+L)S\big)$ | $\mathcal{O}\big((F+L)S\big)$ |
| VarOut | $\mathcal{O}\big(2(F+L)S + R(W/N)\big)$ | $\mathcal{O}\big(2(F+L)S\big)$ |
| DVI | $\mathcal{O}\big(2F + SL + R(W/N)\big)$ | $\mathcal{O}\big(2F + SL\big)$ |
| EDL | $\mathcal{O}\big(F+L\big)$ | $\mathcal{O}\big(F+L\big)$ |
| BEDL | $\mathcal{O}\big(3F + SL\big)$ | $\mathcal{O}\big(3F + SL\big)$ |
| BEDL-PAC | $\mathcal{O}\big(3F + S(L+R)\big)$ | $\mathcal{O}\big(3F + SL\big)$ |

## 5.7 CONCLUSION

In this chapter, we introduced a method for fitting BNNs via type-II maximum likelihood and for performing Bayesian inference within the framework of evidential deep learning. Employing empirical Bayesian methods for inference and combining it with PAC-bounds for regularization, we achieve higher accuracy and better predictive uncertainty estimates while maintaining scalable inference. Exact inference in a fully Bayesian model such as a GP (c.f. Table 5.1) or Hamiltonian Monte Carlo inference for BNNs (Bui et al., 2016) are known to provide better error rates and test log-likelihood scores, yet their computational demand does not scale well to large networks and data-sets. On the other hand, our method shows strong indicators for improvement in uncertainty quantification and predictive performance compared to other BNN approximate inference schemes with reasonable computational requirements. The BEDL-PAC framework benefits might be fruitful in setups such as model-based deep reinforcement learning, active learning, and data synthesis, where uncertainty quantification is a vital ingredient of the predictor.

## 5.8 FURTHER DETAILS

### 5.8.1 *Extensions on the Proposed Model*

This chapter's main text discusses the fact that the CLT trick renders the expectation and variance terms of a post-activation $h$ analytically tractable, without ever actually deriving these expressions. We pluck this hole in the next subsection, where we demonstrate how to compute them. As was the case for the VBP model (see Chapter 3 for the details), these derivations can be extended for a broad class of ReLU variations. The same holds for the discussion in that chapter of the adaptability to skip and residual connections. Unfortunately, it also holds true for the downsides of problems concerning the

inclusion of currently popular neural network building blocks such as pooling and normalization layers.

Instead of rehashing that discussion a second time again, we will instead derive another alleviation to the normalizing problem that was not applicable to the VBP model. A family of activation functions, known as exponential linear units (ELU) (Clevert et al., 2016) and scaled exponential linear unit (SELU)(Klambauer et al., 2017) lessen the dependence on normalization layers. Since they are no longer piecewise linear as the ReLU family, they cannot adapt to the VBP formulation. However, as we will show below, they still share the analytical tractability of the first two moments with the ReLU family leaving them a viable option for a CLT-based model.

#### 5.8.1.1 *First two Moments of the ReLU Activation*

The proposed model relies heavily on the analytical tractability of the ReLU activation, which was already shown and used by Frey and Hinton (1999), but will be rederived here for completeness as they will mirror the computations required for the ELU activation in the next subsection.

Following the notation from our main discussion, we can describe the pre-activation of the $l$-th layer for the $n$-th data point $\boldsymbol{f}_n^l$ to be approximately normally distributed, with a given mean $\boldsymbol{\mu}_n^l$, and variance $(\boldsymbol{\sigma}_n^l)^2$. Dropping the layer and data point identification to lighten the notation, we can compute the expectation and variance of the $k$-th dimension of the post-activation $h_k = \max(0, f_k)$ as follows.

Using $\Phi(\cdot)$ and $\phi(\cdot)$ to refer to the cdf and pdf of the standard normal distribution respectively, the mean can be derived as

$$\begin{aligned}
\mathbb{E}[h] &= \mathbb{E}_{\mathcal{N}(f|\mu,\sigma^2)}[\max(0,f)] = \int_0^\infty \frac{f}{\sigma}\phi\left(\frac{f-\mu}{\sigma}\right)\mathrm{d}f \\
&= \int_{-\mu/\sigma}^\infty (u\sigma+\mu)\phi(u)\,\mathrm{d}u \qquad \textit{substituting } u = \frac{f-\mu}{\sigma} \\
&= \sigma\int_{-\mu/\sigma}^\infty u\phi(u)\,\mathrm{d}u + \mu\int_{-\mu/\sigma}^\infty \phi(u)du \\
&= \sigma\phi\left(\frac{\mu}{\sigma}\right) + \mu\Phi\left(\frac{\mu}{\sigma}\right),
\end{aligned}$$

using that $\Phi(x) = 1 - \Phi(-x)$ to evaluate the second integral. The variance term can be derived analogously as

$$\begin{aligned}
\mathrm{var}[h] &= \mathbb{E}\left[\max(0,f)^2\right] - (\mathbb{E}[h])^2 \\
&= \int_{-\mu/\sigma}^\infty \left(\mu^2 + 2\mu\sigma u + \sigma^2u^2\right)\phi(u)\,\mathrm{d}u - (\mathbb{E}[h])^2 \\
&= \mu^2\Phi\left(\frac{\mu}{\sigma}\right) + 2\mu\sigma\phi\left(\frac{\mu}{\sigma}\right) + \sigma^2\underbrace{\int_{-\mu/\sigma}^\infty u^2\phi(u)\,\mathrm{d}u}_{\Phi\left(\frac{\mu}{\sigma}\right)-\frac{\mu}{\sigma}\phi\left(\frac{\mu}{\sigma}\right)} - (\mathbb{E}[h])^2 \\
&= (\mu^2+\sigma^2)\Phi\left(\frac{\mu}{\sigma}\right) + \mu\sigma\phi\left(\frac{\mu}{\sigma}\right) - \left(\mathbb{E}[h]\right)^2.
\end{aligned}$$

#### 5.8.1.2 *First two moments of the ELU Activation*

The following derivations are an adaptation of the ReLU results to ELU to provide us with an activation function that induces some normalization. We are again interested in the first two moments for the ELU defined as

$$g(x) \triangleq \begin{cases} x, & x > 0 \\ \alpha(\exp(x) - 1) & x < 0 \end{cases}.$$

SELU, as the name implies is a scaled version of this ELU activation with an additional scaling factor $\lambda$ as

$$\tilde{g}(x) \triangleq \lambda \begin{cases} x, & x > 0 \\ \alpha(\exp(x) - 1) & x < 0 \end{cases}.$$

While ELU treats the $\alpha$ as a free hyperparameter, SELU determines $\lambda$ and $\alpha$ numerically to try to ensure that the post-activation is normalized again. See Klambauer et al. (2017) for the specific details. The following derivations treat the general ELU case. The scaled version follows analogously.

Throughout the derivation, we will use the same notation as in the ReLU derivations, i.e. $\boldsymbol{f}$ to refer to an approximately normally distributed pre-activation (with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$) and $\boldsymbol{h} = g(\boldsymbol{f})$ as the corresponding post-activation. We will again suppress the layer and data point indices and focus on a single dimension.

Focusing on the expectation first, we have for the expectation of $h$, that

$$\mathbb{E}[h] = \int_{-\infty}^{0} \alpha(\exp(f) - 1)\mathcal{N}(f|\mu, \sigma^2)\,\mathrm{d}f + \mathbb{E}[\texttt{relu}(f)].$$

$\texttt{relu}(f) \triangleq \max(0, f)$

The first term can be split into

$$\alpha \int_{-\infty}^{0} \exp(f)\mathcal{N}(f|\mu, \sigma^2)\,\mathrm{d}f - \alpha \int_{-\infty}^{0} \mathcal{N}(f|\mu, \sigma^2)\,\mathrm{d}f.$$

We get for these two terms that the first is equal to

$$\alpha \int_{-\infty}^{0} \exp(f)\mathcal{N}(f|\mu, \sigma^2)\,\mathrm{d}f = \alpha \exp(\mu + \sigma^2/2)\Phi\left(-\frac{\mu + \sigma^2}{\sigma}\right)$$

and the second gives

$$\alpha \int_{-\infty}^{0} \mathcal{N}(f|\mu, \sigma^2)\,\mathrm{d}f = \alpha\Phi\left(-\frac{\mu}{\sigma}\right) = \alpha\left(1 - \Phi\left(\frac{\mu}{\sigma}\right)\right).$$

Combining all of this we end up with

$$\mathbb{E}[h] = \alpha\left(\exp(\mu + \sigma^2/2)\Phi\left(-\frac{\mu + \sigma^2}{\sigma}\right) - \Phi\left(-\frac{\mu}{\sigma}\right)\right) + \mathbb{E}[\texttt{relu}(f)].$$

For the variance we have the general form

$$\mathrm{var}[h] = \mathbb{E}\left[h^2\right] - \mathbb{E}[h]^2$$

in which only the first term, i.e. the second moment, is missing. It decomposes again as in the case of the expectation into two terms

$$\mathbb{E}\left[h^2\right] = \int_{-\infty}^{0} \alpha^2(\exp(f)-1)^2 \mathcal{N}(f|\mu,\sigma^2)\,\mathrm{d}f + \mathbb{E}\left[\mathtt{relu}(f)^2\right],$$

of which the second is known from the ReLU derivations. The first term of this equation finally expands into the following monstrosity

$$\begin{aligned}
& \alpha^2 \int_{-\infty}^{0} (\exp(2f) - 2\exp(f) + 1)\mathcal{N}(f|\mu,\sigma^2)\,\mathrm{d}f \\
= \;& \alpha^2 \int_{-\infty}^{-\frac{\mu}{\sigma}} \left(\exp(2\mu + 2\sigma y) - 2\exp(\mu+\sigma y) + 1\right)\phi(y)\,\mathrm{d}y \\
= \;& \alpha^2 \left(\exp(2\mu+2\sigma^2)\Phi\left(-\frac{\mu+2\sigma^2}{\sigma}\right)\right. \\
& \left. -2\exp(\mu+\sigma^2/2)\Phi\left(-\frac{\mu+\sigma^2}{\sigma}\right) + \Phi\left(-\frac{\mu}{\sigma}\right)\right).
\end{aligned}$$

These two moments can be directly used as replacements for the ReLU moments in the proposed model for deeper networks.

#### 5.8.1.3 *Covariance Terms*

Throughout this chapter and the preceding derivations, we have, similar to the VBP discussions, completely dropped covariance terms of the pre/post-activations. Instead, we only computed and propagated essentially a diagonal covariance matrix. Part of the reasoning is shared with the arguments of the discussion there (see Chapter 3.5.3 for details), i.e. a greatly increasing computational cost. However, a difference is that the computation is at least technically doable in the case of the VBP model. This is no longer the case for the CLT-based approach with ReLU activation, such that further approximations are required. Wu et al. (2019) derived such an approximation in their proposed method. However, the runtime cost of a forward pass of the final model increased to about 300 sample-based forward passes, rendering it useless for practically relevant architecture sizes. Additionally, they could show no performance improvements over the diagonalized version.

### 5.8.2 *Experimental Details*

This section contains further details on the hyperparameters of the experiments performed in the main discussion. See `github.com/manuelhaussmann/bedl` for a PyTorch based implementation of the proposed model.

#### 5.8.2.1 *Regression*

The neural net used consists of a single hidden layer of 50 units for all data sets except `protein`, which gets 100. The results for all of the baselines except for Variational Dropout (VarOut) are quoted from the results reported by the respective papers which introduced them. The results on the sparse GP are

reported via Bui et al. (2016). For VarOut, we rely on our own implementation, as there are no official results. BEDL, BEDL-PAC, and VarOut all share the same initialization scheme for the mean and variance parameters for each weight following the initialization of Louizos and Welling (2017), i.e. He-Normal for the means and $\mathcal{N}(-9, 0.001)$ for the log variances. VarOut gets a normal prior with a precision of $\alpha = 1.0$, and all three get an observation precision of $\beta = 100$, to encourage them to learn as much of the predictive uncertainty instead of relying on a fixed hyper-parameter. Note that we keep these values fix and data set independent, different to many of the baselines who set them to data set specific values given cross-validations on separate validation subsets.

Each model is trained with the Adam optimizer (Kingma and Ba, 2015) with default parameters for 100 epochs with a learning rate of $10^{-3}$, with varying minibatch sizes depending on the data set size.

#### 5.8.2.2 *Classification and Out-of-distribution Detection*

The network for this task follows the LeNet5 architecture with the following modifications. Instead of max-pooling layers after the two convolutional layers, the convolutional layers themselves use a larger stride to mimic the behaviour. And for the more complex CIFAR-10 data set, the number of channels in the two convolutional layers is increased from the default 20/50 to 192 each. The number of hidden units for the fully connected layer is increased from 500 to 1000 for that data set following Gal and Ghahramani (2016a).

Since there are no OOD results on the BNN baselines we compare against, we rely on our reimplementations of them, ensuring that they each share the same initialization schemes as in the regression setup. For DVI, we implement the diagonal version and use a sampling-based approximation on the intractable softmax. Each model gets access to five samples whenever it needs to conduct an MC sampling approximation. All models get trained via the Adam optimizer with the default hyperparameters and a learning rate of $10^{-3}$. For EDL, we rely on the public implementation the authors provide and use their hyperparameters to learn the model.

#### 5.8.2.3 *GP variants comparison*

The results for the baselines are taken from the respective original papers. The nets for BEDL and BEDL-PAC consist of two convolutional layers with 96 filters of size $5 \times 5$ and a stride of 5. They are trained until for 50 epochs using Adam with default hyperparameters and a learning rate of $10^{-3}$.

# 6

# LEARNING PARTIALLY KNOWN STOCHASTIC DYNAMICS WITH EMPIRICAL PAC-BAYES

In many engineering applications, it is often easy to model dominant characteristics of a dynamical environment by a system of differential equations with a small set of state variables. In contrast, black-box machine learning methods are often highly accurate but less interpretable. Pushing the model towards high fidelity by capturing intricate properties of the environment, however, often requires highly flexible, over-parameterized models. Fitting these models to data can, in turn, result in over-fitting and hence low generalization ability due to their high capacity.

Neural Stochastic Differential Equations are such a black-box method to model a dynamical environment with neural nets assigned to their drift and diffusion terms. The high expressive power of their nonlinearity comes at the expense of instability in identifying the large set of free parameters.

This chapter combines the benefits of both types of models by hybrid modelling: We set up the learning task as a non-linear system identification problem with partially known system characteristics. In the following, we assume to have access to a differential equation system that describes the dynamics of the target environment with low fidelity, for example, by describing the vector field on a reduced dimensionality, by ignoring detailed models of some system components, or by avoiding certain dependencies for computational feasibility. We incorporate the ODE system provided by the domain expert into a non-linear system identification engine to cover a large scope of dynamical systems resulting in a hybrid model, which we will refer to as a *Bayesian Neural Stochastic Differential Equation (BNSDE)* model. We improve the predictive accuracy of such BNSDEs in three steps:

i) accounting for epistemic uncertainty by assuming probabilistic weights,

ii) principled incorporation of partial knowledge on the state dynamics,

iii) training the resultant hybrid model by an objective derived from a PAC-Bayesian generalization bound.

In our experiments, we observe that this recipe effectively translates partial and noisy prior knowledge into an improved model fit.

This chapter is based on Haußmann et al. (2021). In that publication, Sebastian Gerwinn contributed most of the PAC bound derivation for the proof of Theorem 1. Barbara Rakitsch provided the computational cost analysis. The experimental code uses an implementation of the Euler-Maruyama method by Andreas Look.

To do this, we propose a new algorithm for stable and effective training of such a hybrid BNSDE that combines the strengths of two statistical approaches, which we already got to know in the last chapter: Bayesian model selection (Rasmussen and Williams, 2006), and Probably Approximately Correct Bayesian (PAC-Bayesian) bounds (McAllester, 1999; Seeger, 2002). We improve the theoretical links between these two approaches (Germain et al., 2016) by demonstrating how they can co-operate during training. To this end, we propose a novel training objective that suits SDE inference and derive a PAC-Bayesian generalization bound. Further, we provide a proof that this bound is upper bounded by the marginal likelihood of the BNSDE hyperparameters and a complexity penalizer. Gradients of this upper bound are tied to the actual PAC bound. Hence tightening the upper bound also tightens the PAC bound. Consequently, optimizing this bound amounts to empirical Bayes stabilized by a regularizer developed from first principles. We will refer to using this objective for training as *Empirical PAC-Bayes*.

We demonstrate that our method can translate coarse descriptions of the true underlying dynamics into a consistent forecasting accuracy increase. We first show the necessity of each of the multiple steps that comprise our method in an ablation study. Finally, we demonstrate in a real-world motion capture modelling task that our method outperforms black-box system identification approaches (Chen et al., 2018a; Hegde et al., 2019; Look and Kandemir, 2019) and alternative hybridization schemes that incorporate second-order Newtonian mechanics (Yildiz et al., 2019).

## 6.1 BACKGROUND

Our contribution combines approaches from stochastic differential equations, PAC-Bayes, and empirical Bayes. Apart from stochastic differential equations we have met them already before so we will keep the summary short and refer the reader to the background chapter and Chapter 5 for further details.

### 6.1.1 *Stochastic Differential Equations*

Stochastic differential equations (SDEs) are an extension of ordinary differential equations (ODEs) that also include stochastic fluctuations in the dynamics (Øksendal, 1992). If we let $\boldsymbol{h}_t \in \mathbb{R}^P$ denote the $P$-dimensional state as some time $t$, the dynamics can be written in the following form:

$$\mathrm{d}\boldsymbol{h}_t = f(\boldsymbol{h}_t, t)\,\mathrm{d}t + G(\boldsymbol{h}_t, t)\,\mathrm{d}W_t. \tag{6.1}$$

In this equation the drift term is specified by an arbitrary non-linear function $f(\cdot,\cdot) : \mathbb{R}^P \times \mathbb{R}_+ \to \mathbb{R}^P$ and and the diffusion dynamics are governed by the matrix valued function $G(\cdot,\cdot) : \mathbb{R}^P \times \mathbb{R}_+ \to \mathbb{R}^{P\times P}$. Finally, $W_t$ denotes a $P$-dimensional Wiener Process determining the stochastic fluctuations. The solution to the SDE is a stochastic process $\boldsymbol{h}_t$. By setting the diffusion term $G(\cdot,\cdot)$ to zero, the SDE reverts to an ODE.

As analytical solutions of SDEs, i.e. descriptions of $\boldsymbol{h}_t$ are not available except for specific choices of $f(\cdot,\cdot)$ and $G(\cdot,\cdot)$, one has to resort to numerical approximation methods. Analogous to the practice for ODEs, a common approach is to use the Euler-Maruyama (EM) method (Särkkä and Solin, 2019), which discretizes the SDE into $K$ time steps $t_1,\ldots,t_K$, resulting in the following sample based approximation to the joint distribution:

$$\begin{aligned} \boldsymbol{h}_{t_{k+1}} &= \boldsymbol{h}_{t_k} + f(\boldsymbol{h}_{t_k}, t_k)\Delta t_k + G(\boldsymbol{h}_{t_k}, t_k)\Delta W_k, \\ \Delta W_k &\sim \mathcal{N}(0, \Delta t_k \mathbb{1}_P), \quad \Delta t_k \triangleq t_{k+1} - t_k, \end{aligned} \tag{6.2}$$

where $\mathbb{1}_P$ is a $P$ dimensional identity matrix. Using this sampling scheme, we obtain an approximation to the joint distribution $p(\boldsymbol{h}_{t_1},\ldots\boldsymbol{h}_{t_K})$ for the given (fixed) drift and diffusion functions. In the following, we rely on this approximation scheme.

### 6.1.2 *PAC-Bayes Learning*

Probably Approximately Correct (PAC) bounds quantify the generalization capabilities of a model from a training set to the true data distribution (McAllester, 1999, 2003). To this end, a risk $R(h) \triangleq \mathbb{E}_x\left[l(x,h(x))\right]$ of a hypothesis $h$ is defined via a loss function $l(x,h(x))$ that measures the loss of the hypothesis evaluated at a data point $x$.

Particularly, we build upon the PAC-Bayes formulation, in which the generalization performance of a Bayesian posterior, i.e. a distribution $Q$ over hypotheses, is characterized by the following generic bound:

$$\forall Q: \quad \mathbb{E}_{h\sim Q}\left[R(h)\right] \leq \mathbb{E}_{h\sim Q}\left[R_{\mathcal{D}}(h)\right] + \mathcal{C}(P,Q,\delta,N).$$

It bounds the desired true, but in practice, inaccessible, unknown expected risk $\mathbb{E}_Q\left[R(h)\right]$ by its empirical counterpart in which the risk is averaged across an observed data set $\mathcal{D}$, $\mathbb{E}_Q\left[R_{\mathcal{D}}(h)\right] \triangleq \mathbb{E}_Q\left[\frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} l\big(x,h(x)\big)\right]$. The second term in the bound is a complexity term $\mathcal{C}(P,Q,\delta,N)$, depending on several parameters. A distribution $P$ over the hypothesis independent of the observed data serves as a prior. Additionally, it depends on the number of observed data points $N$ and a confidence variable $\delta$ specifying the probability with which the bound holds (McAllester, 1999; Maurer, 2004).

### 6.1.3 *Empirical Bayes*

Bayesian models define a prior distribution $p_\phi(\theta)$ over parameters $\theta$ with hyperparameter $\phi$, which together with the likelihood $p(\mathcal{D}|\theta)$ defines the full model. The standard approach consists of learning a posterior over these parameters $p(\theta|\mathcal{D})$ while keeping the hyperparameters $\phi$ fixed and marginalizing over $\theta$ in a second step to get the posterior predictive. An alternative approach, which we already discussed and used at length in the last chapter, known as empirical Bayes or type-II maximum likelihood (see e.g. Bishop,

2006), directly marginalizes over the prior, and optimizes the resulting marginal likelihood/prior predictive with respect to the hyperparameters $\phi$, i.e.

$$\arg\max_{\phi} \int p(\mathcal{D}|\theta) p_\phi(\theta) \, \mathrm{d}\theta.$$

## 6.2 THE PROPOSED METHOD

This section describes how to combine these tools into a coherent method and how to perform effective inference with it. We first construct a Bayesian neural SDE and equip it with domain-specific prior knowledge. We then derive a PAC-Bayesian objective to be fitted to data and conclude with results on the proposed model's convergence. Figure 6.1 summarizes the pipeline.

### 6.2.1 *A Hybrid BNSDE*

Application of deep learning to differential equation modelling paves the way to high-capacity predictors for capturing complex dynamics (Chen et al., 2018a; Rackauckas et al., 2020). Neural Stochastic Differential Equations (NSDEs) (Look and Kandemir, 2019; Tzen and Raginsky, 2019) are SDEs as defined in (6.1) where the drift function and potentially also the diffusion function are modelled as neural nets. As an initial step towards effective training, we introduce a prior distribution $p_\phi(\theta)$, parameterized by $\phi$ on the weights $\theta$ of a NSDE drift network, and arrive at

*Bayesian Neural SDE*

$$\mathrm{d}\boldsymbol{h}_t = f_\theta(\boldsymbol{h}_t, t) \, \mathrm{d}t + G(\boldsymbol{h}_t, t) \, \mathrm{d}W_t, \quad \theta \sim p_\phi(\theta), \tag{6.3}$$

which we refer to as a *Bayesian Neural Stochastic Differential Equation (BSNDE)* throughout the rest of the chapter. The epistemic uncertainty introduced on the network weights allows the model to quantify the model uncertainty, i.e. the knowledge of which synaptic map fits best to data, in addition to the aleatoric uncertainty that is modeled by the Wiener Process. For technical reasons clarified below, we have to assume $f_\theta(\cdot,\cdot)$ and $G(\cdot,\cdot)$ to be $L$-Lipschitz-continuous, and $G(\cdot,\cdot)$ not to have any learnable parameters. Two minor constraints, which can also be dropped in practice if one is willing to rely primarily on the empirical performance and neglect the theoretical guarantees they provide.

*Incorporation of prior knowledge*

In real-world applications, a coarse description of the environment dynamics is sometimes available as an incomplete set of differential equations. For instance, the dynamics of a three-dimensional volume might be modeled as a flow through a single point, such as the center of mass. Alternatively, a model on a subset of the system components might be provided. Here, we assume this prior knowledge to be available as an ODE

$$\mathrm{d}\boldsymbol{h}_t = r_\xi(\boldsymbol{h}_t, t) \, \mathrm{d}t, \tag{6.4}$$

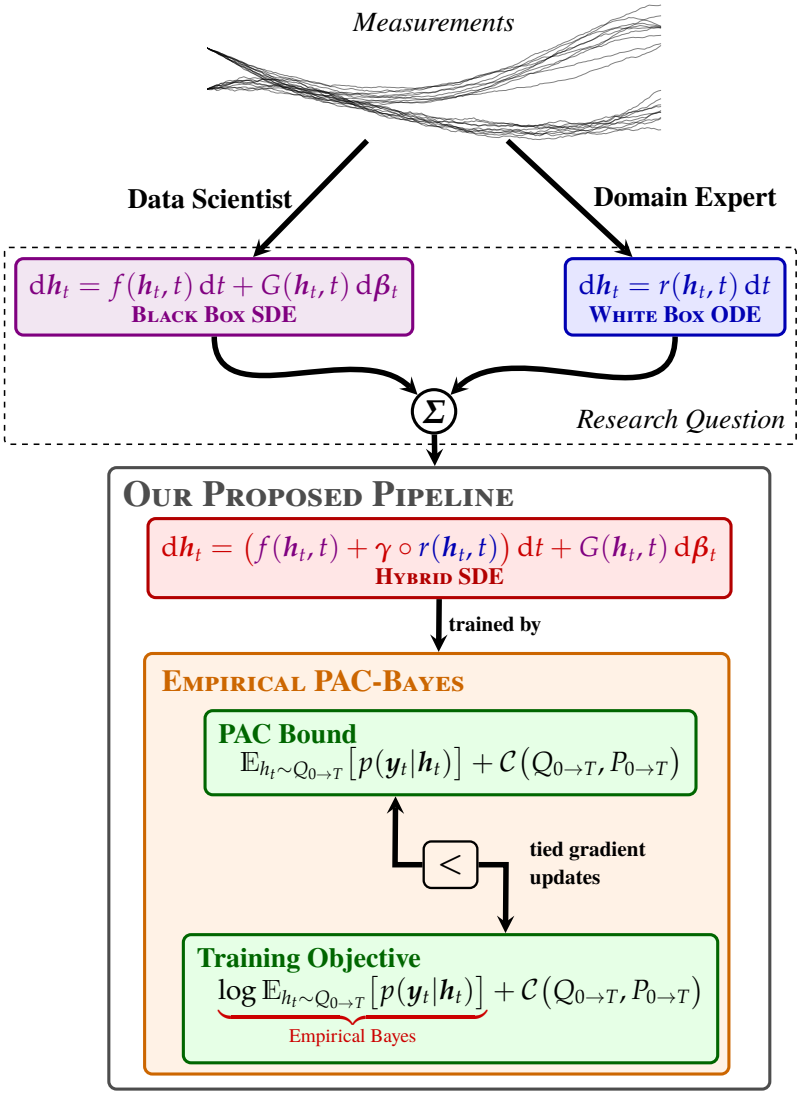



**Figure 6.1: Illustration of the research question we pose and our proposed solution.** Given some measurements, a data scientist can derive a flexible neural network-based black-box approach, while a domain expert can design a more principled but more rigid model based on prior knowledge. The hybrid SDE combines these two approaches, and the PAC-Bayes bound gives a principled approach to learning the parameters of the joint model.

where $r_\xi(\cdot,\cdot) : \mathbb{R}^P \times \mathbb{R}_+ \to \mathbb{R}^P$ is an arbitrary non-linear function parameterized by a fixed set of parameters $\xi$. We can incorporate these known dynamics into the BNSDE by adding them to the drift as follows:

$$\mathrm{d}\boldsymbol{h}_t = \big(f_\theta(\boldsymbol{h}_t, t) + \boldsymbol{\gamma} \circ r_\xi(\boldsymbol{h}_t, t)\big)\,\mathrm{d}t + G(\boldsymbol{h}_t, t)\,\mathrm{d}W_t, \tag{6.5}$$

which can be viewed as a hybrid SDE with the free parameter vector $\boldsymbol{\gamma} \in [0,1]^P$ governing the relative importance of prior knowledge on the learning problem and $\circ$ referring to element-wise multiplication. Although we specified (6.4) within the same dimensional state space as (6.5), $\boldsymbol{\gamma}$ allows us to provide only partial information. When prior knowledge is available only for a subset of the state space dimensions, the remaining dimensions $d$ can be filled in by simply setting $\gamma_d = 0$ as we will demonstrate in the experiments.

We define a prior stochastic process representing solely the prior knowledge of the dynamics as

$$\mathrm{d}\boldsymbol{h}_t = \big(\boldsymbol{\gamma} \circ r_\xi(\boldsymbol{h}_t, t)\big)\,\mathrm{d}t + G(\boldsymbol{h}_t, t)\,\mathrm{d}W_t. \tag{6.6}$$

This prior SDE will be used as a reference distribution for complexity penalization as part of the final PAC-Bayes training objective of our hybrid SDE. Note that we have used the same diffusion term as in (6.5) for specifying the prior SDE, which makes the complexity term within the PAC-formulation tractable, as we will show later. Also, note that $\boldsymbol{\gamma}$ is a free parameter of the prior.

### 6.2.2 *Learning via Empirical Bayes*

Solving the SDE in (6.5) even for fixed parameters $\theta$ over a time interval $[0, T]$ is analytically intractable for basically all practically interesting use cases. While our method is applicable to any discretization scheme, we demonstrate its use with the straightforward EM for simplicity, which gives us a discrete-time version of the hybrid BNSDE as

$$\begin{aligned}
\theta &\sim p_\phi(\theta), \\
\boldsymbol{h}_0 &\sim p(\boldsymbol{h}_0), \\
\boldsymbol{h}_{k+1}|\boldsymbol{h}_k, \theta &\sim \mathcal{N}\left(\boldsymbol{h}_{k+1}\middle|\boldsymbol{h}_k + d(\boldsymbol{h}_k, t_k)\Delta t_k, \Sigma_k\right), \\
d(\boldsymbol{h}_k, t_k) &\triangleq f_\theta(\boldsymbol{h}_k, t_k) + \boldsymbol{\gamma} \circ r_\xi(\boldsymbol{h}_k, t_k),
\end{aligned}$$

with $\Sigma_k \triangleq G(\boldsymbol{h}_k, t_k)G(\boldsymbol{h}_k, t_k)^\top \Delta t_k$, and $\Delta t_k \triangleq t_{k+1} - t_k$. The distribution $p(\boldsymbol{h}_0)$ is an assumption on how the initial states are distributed.

Analogously to latent state space models, we assume that the observations of the latent dynamics described in the three variations, the BNSDE (6.3), its hybrid version (6.5), and the prior stochastic process (6.6) are linked via a likelihood $p(\boldsymbol{y}_k|\boldsymbol{h}_k)$. Specifically, we observe these dynamics as time series $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K\}$ consisting of $K$ $D$-dimensional observations $\boldsymbol{y}_k \in \mathbb{R}^D$, collected at potentially irregularly spaced time points $\boldsymbol{t} = \{t_1, \ldots, t_K\}$. Figure 6.2 gives the plate diagram corresponding to this generative model.

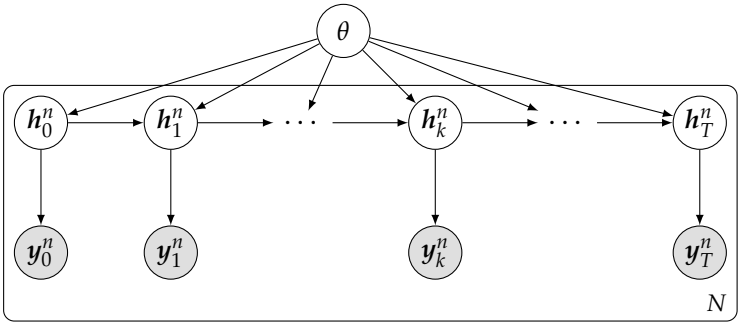



**Figure 6.2:** Plate diagram of the BNSDE after applying the EM discretization.

Given an observed set of $N$ such time series trajectories $\mathcal{D} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_N\}$, the classical Bayesian approach (MacKay, 2003; Gelman et al., 2013) would now require as a first step the inference of the posterior over both the global variables $\theta$ as well as the local variables $\boldsymbol{H}_n = \{\boldsymbol{h}_1^n, ..., \boldsymbol{h}_K^n\}$, i.e. of the distribution $p(\theta, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_N|\mathcal{D})$, and as a second step a marginalization over this posterior to get the posterior predictive. As an analytical solution is intractable, approximate solutions such as Markov Chain Monte Carlo (MCMC) methods or Variational Inference (VI) are required. The application of either of these approaches to BNSDEs is impractical, however. The former due to computational costs, the latter in terms of expressiveness since existing work makes strong independence and structural assumptions on the approximate posterior.

We propose to apply model selection as an alternative path to BNSDE inference. Instead of performing the posterior inference on the latent variables, we marginalize them out and learn those hyperparameters $\phi$ from data that provide the highest log marginal likelihood (Rasmussen and Williams, 2006). That is our BNSDE learns the optimal $\phi^*$ via

$$\arg\max_{\phi} \int p(\mathcal{D}|\boldsymbol{H}) p(\boldsymbol{H}|\theta) p_\phi(\theta) \, \mathrm{d}(\boldsymbol{H}, \theta). \tag{6.7}$$

An advantage of this construction is that the marginal likelihood has an identical functional form to the predictive distribution, which is the quantity of interest in a typical prediction task. Marginal likelihood learning has been applied before in neural networks, for example, as an integral part of the proposed approach discussed in the last chapter. Fitting the hyperparameters of an SDE to data via marginal likelihood maximization can also be viewed as an instance of the simulated likelihood method (Särkkä and Solin, 2019).

Marginalizing over $\theta$ in (6.7) is intractable for most practical use cases. However, it can be approximated by Monte Carlo integration without constructing chains on the global parameters. Sampling directly from the prior, we get for a single observation $n$ and $s = 1, \ldots, S$

$$\begin{aligned} \theta^s &\sim p_\phi(\theta), \\ \boldsymbol{H}^s &\sim p(\boldsymbol{H}|\theta^s), \\ \phi^* &\triangleq \arg\max_{\phi} \log\Big(\frac{1}{S}\sum_{s=1}^{S} p(\mathcal{D}|\boldsymbol{H}^s)\Big). \end{aligned} \tag{6.8}$$

To maximize this objective, we require an efficient computation of gradients with respect to the hyperparameter $\phi$. Access to $\phi$ is only given via samples from the distribution it is parameterizing. In our experiments, we assume this distribution $p_\phi(\theta)$ to be normal, allowing us to use the standard reparameterization. We separate the sampling process into a parameter-free source of randomness and a parametric transformation, i.e. we have $\varepsilon \sim p(\varepsilon), \theta = g_\phi(\varepsilon)$, for a suitable function $g_\phi(\cdot)$. To further reduce the variance noise introduced to the gradients due to this sampling step, we also use the *local reparameterization trick* (Kingma et al., 2015) in the drift, i.e. we sample the layer outputs during the forward propagation instead of individual layer weights, as discussed before.

The objective (6.8) is agnostic to the specific SDE employed. Therefore, we refer to the discretized black-box SDE in (6.3) governing $p(\boldsymbol{H}|\theta)$ and trained with respect to $\phi$ via this objective as *E-Bayes* in the experiments. Analogously, we refer to training a hybrid SDE, i.e. one that incorporates the prior knowledge as in (6.5), with the same method as *E-Bayes-Hybrid*.

*E-Bayes*

*E-Bayes-Hybrid*

### 6.2.3 *A Trainable PAC Bound*

A major downside of the objective in (6.8), when applied to BNSDEs, is that it optimizes a large set of hyperparameters, i.e. the means and variances of the drift network weights, without a proper regularization aside from the implicit regularization inherent in the chosen architecture and the marginalization itself. While the hybrid approach already allows us to incorporate prior expert knowledge, it remains only a guiding signal instead of an explicit model capacity regularizer. Next, we address this problem by developing a training objective derived from a PAC-Bayesian bound that combines the benefits from the results we arrived at so far with a proper regularization scheme.

The proposed approach is still agnostic to the chosen discretization scheme. We therefore refer, to remain general, for any time horizon $T > 0$ to all local latent variables by $\boldsymbol{h}_{0\to T}$. To distinguish the density given by the hybrid SDE in (6.5) from the prior SDE in (6.6), we further refer to the two densities induced by them respectively as $p_{\text{hyb}}(\boldsymbol{h}_{0\to T}|\theta)$ and $p_{\text{pri}}(\boldsymbol{h}_{0\to T})$. This means that we have define two distributions $Q$ and $P$ over the random variables $(\boldsymbol{h}_{0\to T}, \theta)$. For the former, we have the joint distribution of the hybrid process in (6.5)

$$Q_{0\to T}(\boldsymbol{h}_{0\to T}, \theta) \triangleq p_{\text{hyb}}(\boldsymbol{h}_{0\to T}|\theta)p_\phi(\theta), \tag{6.9}$$

while the latter stands for the joint distribution of the prior process in (6.6)

$$P_{0\to T}(\boldsymbol{h}_{0\to T}, \theta) \triangleq p_{\text{pri}}(\boldsymbol{h}_{0\to T})p_{\text{pri}}(\theta). \tag{6.10}$$

Although the prior process is independent of the drift parameters $\theta$, we specify a fixed prior distribution $p_{\text{pri}}(\theta)$ within the prior $P$, which we choose to be a standard normal density within our experiments. To be compliant with the notational practice in the PAC Bayesian literature, we denote the prior distribution as $P$ and the posterior distribution that is fit to data as $Q$. In the PAC-Bayesian

framework, $P$ and $Q$ do not have to be linked to each other via application of the Bayes rule on an explicitly defined likelihood. Instead, they can be any pair $(P, Q)$ of data-independent and data-dependent distributions over the family of hypothesis.

As both $Q$ and $P$ share the same diffusion term, the Kullback-Leibler (KL) divergence between these processes can be calculated by extending the proof of Archambeau et al. (2008). The following Lemma holds for any choice of diffusion term $G(\cdot, \cdot)$. The proofs for the Lemma and the following Theorems can be found at the end of this chapter in Section 6.6.

LEMMA 1. *For the process distributions $Q_{0\to T}$ and $P_{0\to T}$ as defined above, it holds that*

$$\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right) = \frac{1}{2}\int_0^T \mathbb{E}_{Q_{0\to T}}\left[f_\theta(\boldsymbol{h}_t, t)^\top \mathbf{J}_t^{-1} f_\theta(\boldsymbol{h}_t, t)\right] \mathrm{d}t + \mathrm{KL}\left(p_\phi(\theta) \parallel p_{\mathrm{pri}}(\theta)\right),$$

*for some $T > 0$, where $\mathbf{J}_t = G(\boldsymbol{h}_t, t)G(\boldsymbol{h}_t, t)^\top$.*

This Lemma provides one of the main ingredients for deriving a PAC-Bayesian bound on the generalization performance of a learned distribution $Q_{0\to T}$. To derive such a bound, we additionally specify the risk via a loss function measuring the model mismatch. We assume the likelihood function $p(\boldsymbol{y}_t|\boldsymbol{h}_t)$ to be uniformly bounded everywhere. In our experiments, we ensure this condition by choosing the likelihood to be a normal density with bounded variance, i.e. bounded mass on the mode. We then define the true risk of a draw from $Q_{0\to T}$ on an i.i.d. sampled trajectory $\boldsymbol{y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K\}$ at discrete and potentially irregular time points $t_1, \ldots, t_K$ drawn from an unknown ground-truth stochastic process $\mathfrak{G}(t)$ as the expected model misfit on the sample. Specifically, we define the risk of a specific hypothesis$H \in \mathcal{H} = (\boldsymbol{h}_{0\to T}, \theta)$ as follows:

$$R(H) \triangleq \mathbb{E}_{\boldsymbol{Y}_k \sim \mathfrak{G}(t)}\left[1 - \frac{1}{\overline{B}_K}\prod_{k=1}^{K} p(\boldsymbol{y}_k|\boldsymbol{h}_k)\right], \tag{6.11}$$

for time horizon $T > 0$, and $\overline{B}_K$ defined as

$$\overline{B}_K \triangleq \max_{\boldsymbol{y}_k, \boldsymbol{h}_k} \prod_{k=1}^{K} p\left(\boldsymbol{y}_k|\boldsymbol{h}_k\right) \leq \left(\max_{\boldsymbol{y}_k, \boldsymbol{h}_k} p\big(\boldsymbol{y}_k|\boldsymbol{h}_k\big)\right)^K.$$

That is, we consider one minus the marginal likelihood of the data rescaled to $[0, 1]$, ensuring that a hypothesis giving a high marginal probability gives a low overall risk. The rescaling to $[0, 1]$ is due to technical requirements in the derivation of the following theorem.

The corresponding empirical risk on a data set consisting of $N$ trajectories, $\mathcal{D} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, is then analogously defined as

$$R_{\mathcal{D}}(H) \triangleq \frac{1}{N}\sum_{n=1}^{N}\left(1 - \frac{1}{\overline{B}_K}\prod_{kn=1}^{K} p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^n)\right). \tag{6.12}$$

Next, we develop a PAC-Bayesian generalization bound building on these risk definitions. Furthermore, we upper bound it with a trainable objective.

THEOREM 1. *The expected true risk is bounded above with probability* $\mathbb{P} \geq 1 - \delta$ *by:*

$$\begin{aligned}
&\mathbb{E}_{H\sim Q_{0\to T}}\left[R(H)\right] \\
&\qquad \leq \mathbb{E}_{H\sim Q_{0\to T}}\left[R_{\mathcal{D}}(H)\right] + \mathcal{C}_{\delta}(Q_{0\to T}, P_{0\to T}) && (6.13)\\
&\qquad \leq -\frac{1}{N}\sum_{n=1}^{N}\log\left(\frac{1}{S}\sum_{s=1}^{S}\prod_{k=1}^{K} p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^{n,s})\right) \\
&\qquad\quad + \underbrace{\mathcal{C}_{\delta/2}(Q_{0\to T}, P_{0\to T}) + \sqrt{\frac{\log(2N/\delta)}{2S}} + K\log\overline{B}}_{\triangleq C} && (6.14)\\
&\qquad \leq -\frac{1}{SN}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{k=1}^{K}\log\left(p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^{n,s})\right) + C && (6.15)
\end{aligned}$$

*where* $\overline{B} \triangleq \max_{\boldsymbol{y}_k,\boldsymbol{h}_k} p(\boldsymbol{y}_k|\boldsymbol{h}_k)$ *is the uniform bound,* $S$ *is the sample count taken independently for each observed sequence, and the complexity functional is given for some* $\delta > 0$ *as*

$$\mathcal{C}_{\delta}(H_{0\to T}, P_{0\to T}) = \sqrt{\frac{\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right) + \log(4\sqrt{N}/\delta)}{2N}}$$

*with* $\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right)$ *as in Lemma 1.*

Theorem 1 can be used to learn a posterior distribution $Q_{0\to T}$ from data by adjusting $\phi$ (hidden in the notation). Additionally, we could also learn the importance of the prior by fitting the $\gamma$ parameter to data. While directly learning $\gamma$ by optimizing the PAC-bound violates the generalization guarantee, we can define a collection of prior distributions $P_{0\to T}$ for a set $\Gamma$ of discretized values of $\gamma$ and employ the same union bound as Reeb et al. (2018). The resulting PAC-bound differs by a constant accounting for the number of distinct $\gamma$ values within the collection. Therefore, we can use the same gradient based optimization to learn $\gamma$ and quantize the value to the closest point within $\Gamma$ to evaluate the PAC bound. However, throughout the experiments, we will always rely on fixed choices for $\gamma$.

### 6.2.4 *The Training Algorithm*

The first term in (6.13) does not correspond to the empirical Bayes objective as it averages over likelihoods, not log-likelihoods (Germain et al., 2016). However, the first term in (6.14) provides a sampling-based approximation to the empirical Bayes objective. By defining the risk in such a way and employing the PAC-Bayesian framework, we obtain a regularized version of empirical Bayes. Although placing the $\log(\cdot)$ function into its summands loosens the bound on the true risk, it improves numerical robustness and optimizing (6.15)

still tightens the original PAC-Bayesian bound, i.e. (6.13), as shown in the following corollary.

COROLLARY 1. *For Lipschitz-continuous risk and likelihood, a gradient step that reduces* (6.15) *also tightens the PAC bound in* (6.13).

*Final Objective to be optimized*

Minimizing (6.15) hence closes the loop as the empirical Bayes objective derived in (6.8) reappears in (6.14) but is now combined in a principled way with the regularizing $\mathcal{C}_\delta$. We can ignore the terms that do not depend on $\phi$ and adopt the remaining expression bound as our final objective and learn $\phi^*$ via

$$\begin{aligned}\phi^* \triangleq \arg\max_{\phi} \; & \frac{1}{SN} \sum_{n=1}^{N} \sum_{s=1}^{S} \sum_{k=1}^{K} \log\left(p(\boldsymbol{y}_k^n | \boldsymbol{h}_k^{n,s})\right) \\ & + \sqrt{\left(\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right) + \log(4\sqrt{N}/\delta)\right)/2N}.\end{aligned} \tag{6.16}$$

In this training procedure, we only optimize with respect to $\phi$, which control the drift term via parametrizing the distribution over the weights of the BNN modelling it. To also learn the diffusion, one could represent $G(\cdot,\cdot)$ also by a BNN. However, the corresponding training procedure would invalidate the PAC statement. Nevertheless, one could learn the diffusion term on a held-out data set and then incorporated as fixed to the bound (6.15). As Theorem 1 applies to any diffusion term, we keep the genericness of its statement. However, in the experiments, we stick to a constant diffusion term for practical reasons.

Although we require i.i.d. observations of time series in the theory, we can in practice use mini-batches of trajectories provided that the batches are sufficiently far apart so that they become essentially independent. The objective (6.14) differs from the one in (6.8) only by the complexity term. An algorithmic description of the overall procedure is given in Algorithm 2.

Our sampling-based method naturally couples with the EM approximation and inherits its convergence properties. With the following theorem, we show strong convergence to the true solution with shrinking step size, which can be shown by extending the plain EM proof (Kloeden and Platen, 2011).

THEOREM 2 (STRONG CONVERGENCE). *Let $\boldsymbol{h}_t^\theta$ be an Itô process as in* (6.3) *with drift and diffusion parameters $\theta$ and $\widetilde{\boldsymbol{h}}_t^\theta$ its Euler-Maruyama approximation for some regular step size $\Delta t > 0$.*[1] *For some coefficient $R > 0$ and any $T > 0$, the following inequality holds*

$$\mathbb{E}\left[\left|\mathbb{E}_\theta\left[\boldsymbol{h}_T^\theta\right] - \frac{1}{S}\sum_{s=1}^{S} \widetilde{\boldsymbol{h}}_T^{\theta^{(s)}}\right|\right] \leq R\Delta t^{1/2} \qquad \text{as } S \to \infty,$$

*where $\{\theta^{(s)} \sim p_\phi(\theta) | s = 1,\ldots,S\}$ are i.i.d. draws from a prior $p_\phi(\theta)$.*

1 Here we use the superscript $\theta$ notation to make the dependence on the parameters explicit.

**Algorithm 2:** E-PAC-Bayes-Hybrid Loss

**Input:** set of $N$ trajectories $\mathcal{D}$, prior drift $r_{\tilde{\zeta}}(\cdot,\cdot)$, time points $\boldsymbol{t}$, drift $f_\theta(\cdot,\cdot)$, diffusion $G(\cdot,\cdot)$, weight distribution $p_\phi(\theta)$, number of samples $S$, prior parameter $\gamma$

**Output:** training objective loss

```
// initialize marginal log-likelihood (mll) and kl
```
$\text{mll} \leftarrow 0$

$\text{kl} \leftarrow 0$

**for** $n \in \{1,\dots,N\}$ **do** `// for each trajectory`

  **for** $s \in \{1,\dots,S\}$ **do** `// and each sample`

    `// sample initial state and weights`

    $\boldsymbol{h}_0^{n,s} \sim p(\boldsymbol{h}_0)$

    $\theta^{n,s} \sim p_\phi(\theta)$

    `// for each of the K steps`

    **for** $k \in \{1,\dots,K\}$ **do**

      `// get drift, prior, and diffusion output`

      $f_k^{n,s} \leftarrow f_{\theta^{n,s}}(\boldsymbol{h}_{k-1}^{n,s}, t_{k-1})$

      $r_k^{n,s} \leftarrow r_{\tilde{\zeta}}(\boldsymbol{h}_{k-1}^{n,s}, t_{k-1})$

      $G_k^{n,s} \leftarrow G(\boldsymbol{h}_{k-1}^{n,s}, t_{k-1})$

      `// sample stochasticity`

      $\Delta t_k \leftarrow t_k - t_{k-1}$

      $W_k^{n,s} \sim \mathcal{N}(0, \Delta t_k \mathbb{1})$

      `// update state`

      $\boldsymbol{h}_k^{n,s} \leftarrow \boldsymbol{h}_{k-1}^{n,s} + (f_k^{n,s} + \gamma r_k^{n,s})\Delta t_k + G_k^{n,s} W_k^{n,s}$

      `// and update mll and kl`

      $\text{mll} \leftarrow \text{mll} + \frac{1}{SN} \log p(\boldsymbol{y}_k^n | \boldsymbol{h}_k^{n,s})$

      $\text{kl} \leftarrow \text{kl} + \frac{1}{2S} f_k^{n,s\top} (G_k^{n,s} G_k^{n,s\top})^{-1} f_k^{n,s} \Delta t_k$

    **end**

  **end**

**end**

```
// add penalty for modified drift distribution
```
$\text{kl} \leftarrow \text{kl} + D_{KL}\Big(p_\phi(\theta) || p_{\text{pri}}(\theta)\Big)$

```
// and assign final loss
```
$\text{loss} \leftarrow -\text{mll} + \sqrt{\left(\text{kl} + \log(4\sqrt{N}/\delta)\right)/(2N)}$

```
// to be returned and optimized
```
**return** loss

## 6.3 RELATED WORK

EMPIRICAL BAYES AS PAC-BAYES LEARNING. Germain et al. (2016) propose a learnable PAC-Bayesian bound that provides generalization guarantees as a function of a marginal log-likelihood. Our method differs from this work in two main lines. First, Germain et al. (2016) define risk as $-\log p(\boldsymbol{Y}|\boldsymbol{H}) \in$

$(-\infty, +\infty)$ and compensate for the unboundedness by either truncating the support of the likelihood function or introducing assumptions on the data distribution, such as it being sub-Gaussian or sub-Gamma. As defined in (6.11), our risk is different in that it assumes uniform boundedness yet can be incorporated into a PAC-Bayesian bound without further restrictions on the data. Second, Germain et al. (2016)'s bound is an unparameterized rescaling of the marginal log-likelihood. Hence, it is not linked to a capacity penalizer, which can be used at *training time* for regularization. Applying this method to hybrid sequence modelling boils down to performing plain empirical Bayes, i.e. what we refer to as *E-Bayes* in our experiments.

DIFFERENTIAL GPS. Hegde et al. (2019) model the dynamics of the activation maps of a feed-forward learner by the predictive distribution of a Gaussian Process. This method allocates the mean of a GP as the drift and its covariance as the diffusion. It then infers the resulting model using variational inference. While direct application of this method to time series modeling is not straightforward, we represent it in our experiments by sticking to our generic non-linear BNSDE design from (6.3), and infer it by maximizing the ELBO

$$\mathcal{L}(\phi) = \mathbb{E}_{\boldsymbol{H},\theta}\big[\log\ p(\boldsymbol{Y}|\boldsymbol{H})\big] - D_{KL}\big(p_\phi(\theta)||p(\theta)\big),$$

applying the local reparameterization trick on $\theta$. Although variational inference can be seen from a PAC-perspective by choosing the log-likelihood as the loss (Knoblauch et al., 2019), the ELBO does not account for the deviation of variational posterior over latent dynamics from the prior latent dynamics. We refer to this baseline in the experiments as *D-BNN (VI)*. The approximate posterior design here closely follows the PR-SSM approach (Doerr et al., 2018), which represents state of the art in state-space modelling.

DIFFERENTIAL BNNS WITH SGLD. The learning algorithm of Look and Kandemir (2019) shares our BNSDE modelling assumptions. However, it uses Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011) to infer $\theta$. The algorithm is equivalent to performing the MAP estimation of the model parameters in (6.3) while distorting the gradient updates with decaying normal noise that determines the learning rate.

BLACK-BOX IDENTIFICATION OF DYNAMIC SYSTEMS. There are various approaches to identify a dynamical system that differ in the model class used for fitting the right-hand side of the differential equation and may also allow for transitional noise (see e.g. Brunton et al., 2016; Durstewitz, 2017). These approaches could be incorporated into ours, using their transition likelihood and prior over parameters. In fact, our black-box neural SDE can be seen as one instance of such a black-box identification of dynamical systems. As we are mainly interested in incorporating prior knowledge into such black-box models, we chose as a comparison one such competitor (Hegde et al., 2019), which allows for the most flexible right-hand side with reported results on the CMU Motion capture data set (Tab. 6.2).

## 6.4 EXPERIMENTS

We evaluate the following four variants of our method that we discussed throughout this chapter:

(i) *E-Bayes*. A simple baseline that optimizes the marginal likelihood via empirical Bayes without the incorporation of prior knowledge, i.e. training the generative model in (6.8) with the dynamical system $p(\boldsymbol{h}_{0\to T})$ as given by (6.3). We interpret this variant as applying Germain et al. (2016)'s approach to our BNSDE setup.

(ii) *E-PAC-Bayes.* An empirical PAC-Bayes approach on learning the BNDSE model using the objective in (6.15) with an uninformative prior drift, i.e. with the assumption that $r_\xi(\boldsymbol{h}_t, t) = 0$.

(iii) *E-Bayes-Hybrid.* This model uses the same training objective as the baseline in (i), i.e. a direct optimization of the marginal likelihood, however with the inclusion of prior knowledge in the drift term giving the hybrid model as proposed in (6.5).

(iv) *E-PAC-Bayes-Hybrid*. This final version is the extension of (ii) to the hybrid model (6.5) with the same loss as *E-PAC-Bayes*. It is the combination we propose to use in practice.

In summary, we extend the empirical Bayes objective in (6.8) by PAC-Bayes to tune many hyperparameters without overfitting and incorporate prior domain knowledge in a principled way. See the further details section at the end of this chapter for more details on each of the experiments' setup. [2]

### 6.4.1 *Lotka-Volterra*

We demonstrate the benefits of incorporating prior knowledge although it is a coarse approximation to the true system. We consider the Lotka-Volterra system specified as:

$$\begin{aligned} \mathrm{d}x_t &= (\theta_1 x_t - \theta_2 x_t y_t)\,\mathrm{d}t + 0.2\ \mathrm{d}\beta_t, \\ \mathrm{d}y_t &= (-\theta_3 y_t + \theta_4 x_t y_t)\,\mathrm{d}t + 0.3\ \mathrm{d}\beta_t. \end{aligned}$$

with $\boldsymbol{\theta} = (2.0, 1.0, 4.0, 1.0)$. Assuming that the trajectory is observed on the interval $t = [0, 1]$ with a resolution of $\mathrm{d}t = 0.01$, we compare the following three methods: *i)* the black-box BNSDE without prior knowledge, *ii)* the white-box SDE in (6.6) representing partial prior knowledge (parameters are sampled from a normal distribution centered on the true values with a standard deviation of 0.5), and finally *iii)* combining them in our proposed hybrid method. The outcome is summarized in Figure 6.3. While the plain black-box model delivers a poor fit to data, our hybrid BNSDE brings significant improvement from relevant but inaccurate prior knowledge.

2 See `https://github.com/manuelhaussmann/bnsde` for a reference implementation.

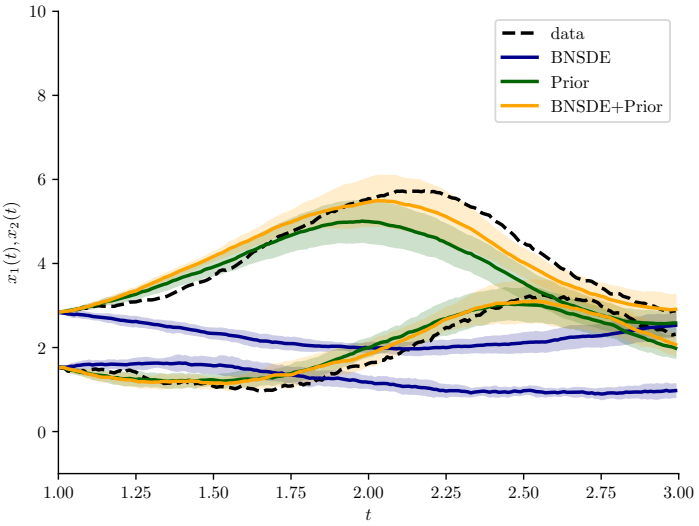



**Figure 6.3: Lotka-Volterra visualization.** Error bars indicate three standard deviations over 10 trajectories starting from the true value at $t = 1$. The predictions over 200 time steps ($dt = 0.01$) are for: *i)* a BNSDE trained without prior knowledge, *ii)* an SDE with known prior parameters, *iii)* the joint hybrid BNSDE. The dashed lines are the observed trajectories for $x_t$ and $y_t$.

### 6.4.2 *Lorenz Attractor*

The classical chaotic non-linear system known as the *Lorenz Attractor* has the the following inherently unsolvable dynamics

$$\begin{aligned} \mathrm{d}x_t &= \zeta(y_t - x_t)\,\mathrm{d}t + \mathrm{d}W_t, \\ \mathrm{d}y_t &= \big(x_t(\kappa - z_t) - y_t\big)\,\mathrm{d}t + \mathrm{d}W_t, \\ \mathrm{d}z_t &= (x_t y_t - \rho z_t)\,\mathrm{d}t + \mathrm{d}W_t, \end{aligned}$$

where $\zeta = 10, \kappa = 28, \rho = 2.67$, and $W_t$ is a random variable following a Wiener process with unit diffusion. We generate 2000 observations from the above dynamics initiating the system at $x_0, y_0, z_0 = (1, 1, 28)$, and use the first half for training and the rest for testing. Figure 6.4 gives the data and visualizes the behaviour of the Lorenz Attractor, demonstrating the qualitative difference between the training and test data.

We split the training and the test data into 20 sequences of length 24, which can be interpreted as i.i.d. samples of the system with different initial states. Table 6.1 gives the 24-step ahead forecasting error in mean squared error (MSE) on the test set for our variants. In each repetition, *E-Bayes-Hybrid* and *E-PAC-Bayes-Hybrid* are provided prior knowledge of one of the three differential equations after distorting the corresponding parameter by standard normally distributed noise. The other equations are hidden by assigning the corresponding dimensions of $\gamma$ to zero. To set up the corresponding prior and model, we used a constant diffusion with $G = \mathbb{1}$. Despite the imprecision of the provided prior knowledge, the largest performance leap comes from the

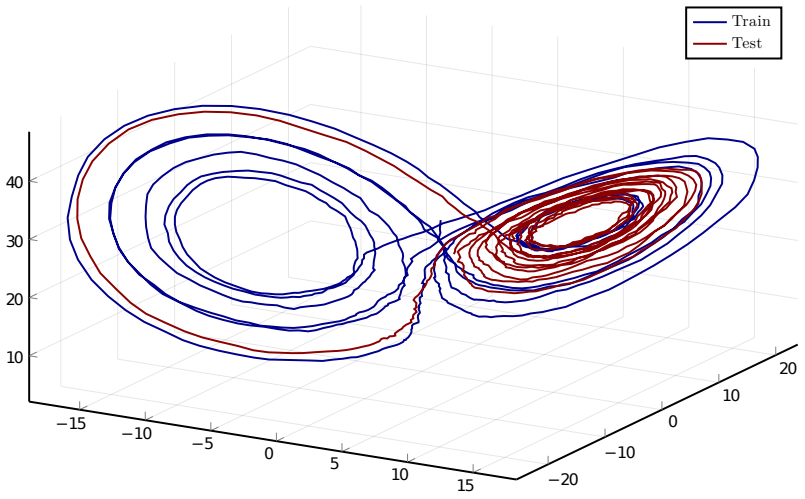



**Figure 6.4: Visualization of the stochastic Lorenz attractor.** Of the 2000 observations, the first 1000 constitute the training data (marked in blue), while the second 1000 are the test observations (marked in red). Note the qualitative difference of the two sets.

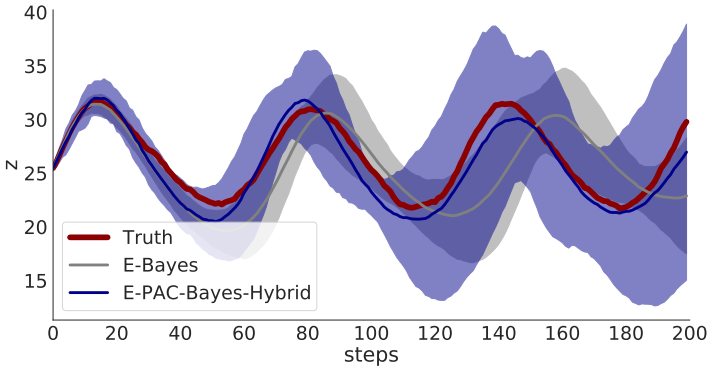



**Figure 6.5:** Predicted trajectory for 200 time steps starting at $T = 10$ of the Lorenz data set mapped to one dimension. The error bars indicate $\pm 2$ standard deviations over 21 trajectories.

hybrid models. The complexity term on the PAC-Bayesian bound restricts the model capacity for black-box system identification while it improves the hybrid setup.

Figure 6.5 visualizes the predicted trajectories on the test sequence for prior knowledge on $\mathrm{d}z_t$. Even with weak prior knowledge, the proposed model is stable longer than the baseline as well as showing a proper increase in the variance as the predicted trajectory increases, while the baseline diverges sooner without a proper increase in uncertainty. See the further details section at the end of this chapter for the corresponding plots of the other dimensions.

**Table 6.1:** Ablation study on the Lorenz attractor to evaluate the contributions of the prior knowledge on the predictive performance measured in mean squared error (MSE) with standard error over fifty repetitions. The hybrid models ((iii), (iv)) consistently improve on the black box models ((i),(ii)). The last row (v) shows the performance for the case the model has full access to the true dynamics with noisy parameters in (6.4).

| Prior Knowledge | Model | Test MSE |
|---|---|---|
| None | (i) | $29.20_{\pm 0.19}$ |
| | (ii) | $29.05_{\pm 0.23}$ |
| $\gamma = [1,0,0], \quad \zeta \sim \mathcal{N}(10,1)$ | (iii) | $27.58_{\pm 0.17}$ |
| | (iv) | $27.42_{\pm 0.16}$ |
| $\gamma = [0,1,0], \quad \kappa \sim \mathcal{N}(28,1)$ | (iii) | $15.87_{\pm 0.46}$ |
| | (iv) | $15.06_{\pm 0.35}$ |
| $\gamma = [0,0,1], \quad \rho \sim \mathcal{N}(2.67,1)$ | (iii) | $27.82_{\pm 0.26}$ |
| | (iv) | $28.37_{\pm 0.21}$ |
| $\gamma = [1,1,1]$, $(\zeta,\kappa,\rho)^\top \sim \mathcal{N}\big((10,28,2.67)^\top, \mathbb{1}_3\big)$ | (v) | $16.40_{\pm 2.31}$ |

### 6.4.3 *CMU Walking Data Set.*

We benchmark against state of the art on this motion capture data set following the setup of Yildiz et al. (2019). We train an *E-PAC-Bayes* model on the `MOCAP-1` data set consisting of 43 motion capture sequences measured from 43 different subjects. The drift net of the learned BNSDE is then treated as weak and broad prior knowledge of human walking dynamics. We use `MOCAP-2` with 23 walking sequences from Subject 35 to represent a high-fidelity subject-specific modelling task. As Yildiz et al. (2019) report in Table 2 of their paper, the state of the art of subject-independent mocap dynamic modelling has twice as high prediction error as subject-specific dynamics (MSE of 15.99 versus 8.09). Analogously to the Lorenz attractor experiment, we fixed the PAC-variants' prior diffusion term to be constant. See Table 6.2 for a summary of the test MSEs and negative log-likelihoods our the four variant approaches on this experiment. Our method delivers the best prediction accuracy and model fit when all its components are active.

### 6.4.4 *Computational Cost*

We close our discussion of the experiments by present the runtimes of the different variations in terms of floating-point operations (FLOPs) of the different approaches in Table 6.3, assuming time series data of length $T$.[3] D-BNN samples the weights of the neural network directly leading to the runtime term $\mathcal{O}(MTF)$. All other approaches do not sample the weights but the linear acti-

3 See the caption of that table for a definition of the parameters discussed in this section.

**Table 6.2:** Benchmarking of our method on the CMU Motion Capture Data Set. Mean Squared Error (MSE) and Negative Log-Likelihood (NLL) on 300 future frames is averaged over ten repetitions ($\pm$ standard deviation).

| **Method** | **Test MSE** | **Test NLL** |
|---|---|---|
| DTSBN-S (Gan et al., 2015) | 34.86±0.02 | Not Applicable |
| npODE (Heinonen et al., 2018) | 22.96 | Not Applicable |
| Neural-ODE (Chen et al., 2018a) | 22.49±0.88 | Not Applicable |
| ODE$^2$VAE (Yildiz et al., 2019) | 10.06±1.40 | Not Reported |
| ODE$^2$VAE-KL (Yildiz et al., 2019) | 8.09±1.95 | Not Reported |
| D-BNN (SGLD) (Look and Kandemir, 2019) | 13.89±2.56 | 747.92±58.49 |
| D-BNN (VI) (Hegde et al., 2019) | 9.05±2.05 | 452.47±102.59 |
| (i) E-Bayes | 8.68±1.56 | 433.76±77.78 |
| (ii) E-PAC-Bayes | 9.17±1.20 | 489.82±67.06 |
| (iii) E-Bayes-Hybrid | 9.25±1.99 | 462.82±99.61 |
| (iv) E-PAC-Bayes-Hybrid | **7.84**±1.41 | **415.38**±80.37 |

**Table 6.3:** Computational cost analysis in FLOPs for time series of length **T**. **M:** Number of Monte Carlo Samples. **W:** Number of weights in the neural net. **F:** Forward pass cost of a neural net. **L:** Cost for computing the likelihood term. **D:** Number of dimensions of the targeted SDE. **P:** Cost of a prior SDE integration.

| **Method** | **Training per Iteration** |
|---|---|
| D-BNN (SGLD) | $\mathcal{O}(MTF + MTDL + W)$ |
| Variational Bayes | $\mathcal{O}(2MTF + MTDL + W)$ |
| E-Bayes | $\mathcal{O}(2MTF + MTDL)$ |
| E-PAC-Bayes | $\mathcal{O}(2MTF + MTDL + W + TMD^3)$ |
| E-Bayes-Hybrid | $\mathcal{O}(2MTF + MTDL + MTP)$ |
| E-PAC-Bayes-Hybrid | $\mathcal{O}(2MTF + MTDL + W + TMD^3 + MTP)$ |

vations of each data points leading to $\mathcal{O}(2MTF)$. When we apply empirical Bayes, we do not use any regularization term on the weights, while all other approaches contain a penalty term with cost $\mathcal{O}(W)$.Using the PAC-framework, we employ a second regularization term that leads to an additional runtime cost of $\mathcal{O}(TMD^3)$.However, the cubic cost in $D$ is invoked by inverting the diffusion matrix $G(h_t, t)$ and can be further reduced by choosing a simpler form for $G(h_t, t)$ (e.g. diagonal as we do throughout the experiments). In case that prior knowledge is available in ODE form, we need to compute the corresponding drift term for each time point and each MC sample leading to the term $\mathcal{O}(MTP)$.

## 6.5 CONCLUSION

We have shown that our method incorporates vague prior knowledge into a flexible Bayesian black-box modelling approach for learning SDEs resulting in a robust learning scheme guided by generalization performance via a PAC-Bayesian bound. The method is easily adaptable to other solvers. For example, the training loss derived in (6.8) can also be optimized using a closed-form normal assumed density scheme applied over a stochastic Runge-Kutta variant (Li et al., 2019). Independent from the sampling scheme and model used, our tied gradient update procedure allows training on the loose, yet numerically stable, bound while providing an improvement with respect to the generalization guarantees on its tighter counterpart. Our stochastic approximation of the data log-likelihood currently relies on samples obtained from the prior, yet could be improved by incorporating a more sophisticated sampling scheme, for example using particle filtering (Kantas et al., 2015). Finally, the bound in (6.15) has the potential to be vacuous for certain drift nets, incorporating a Hoeffding assumption (Alquier et al., 2016) could further tighten it.

### BROADER IMPACT

The proposed approach allows for a principled way of incorporating prior knowledge into the learning stochastic differential equations, together with the flexibility and strength of deep Bayesian neural nets. As such, it shares the potential benefits and risks of both. The growing field of combining differential equations with neural networks has great potential as it allows for a combination of the often disjoint strands of mostly data-driven approaches (deep learning) with mostly symbolic, i.e. model-driven, approaches (differential equations), combining the best of both worlds by offering both the possibility of improved predictive performance as well as greater interpretability. SDEs are agnostic to the task at hand. So the proposed method inherits both their beneficial potential as a powerful tool in many fields of science and their ethically doubtful applications.

It also inherits from deep learning the downsides of potential susceptibility to problems such as adversarial attacks and predictive overconfidence. There is growing literature indicating that the Bayesian approach to neural nets we use seem to be more robust against such problems than standard deterministic nets. Still, most of that is ongoing research without final results. Note that we are not aware of any work showing adversarial attacks in the neural network + differential equation literature, but this should, of course, not be read as a proof of absence. Summarizing the broader impact in one sentence, our proposed approach and other work in this direction have the potential to reinforce the positive and negative influence differential equations are already having, without offering guarantee only benefits. The ethical responsibility remains with the scientist/engineer building on top of it.

## 6.6 FURTHER DETAILS

This section concludes our exploration of BNSDE models with proofs of the claims we made throughout this chapter and further details and results on the three experiments discussed.

### 6.6.1 *Continuous Time SDEs*

Solving the SDE system in (6.1) for a time interval $[0, T]$ and fixed $\theta$ requires computing integrals of the form

$$\int_0^T \mathrm{d}\boldsymbol{h}_t = \int_0^T f_\theta(\boldsymbol{h}_t, t)\,\mathrm{d}t + \int_0^T G(\boldsymbol{h}_t, t)\,\mathrm{d}W_t.$$

This operation is intractable for almost any practically relevant choice of $f_\theta(\cdot,\cdot)$ and $G(\cdot,\cdot)$. The integral around the drift term $f_\theta(\cdot,\cdot)$ does not have an analytical solution, due both to potential nonlinearities of the drift and to the fact that $\boldsymbol{h}_t \sim p(\boldsymbol{h}_t, t)$ is a stochastic random variable following an implicitly defined distribution. Also, the diffusion term involves the Itô integral (Øksendal, 1992) about $W_t$ which multiplies the non-linear function $G(\cdot,\cdot)$.

For each of the SDEs in (6.5) and (6.6), we could alternatively to the Euler-Maruyama integration scheme use the Fokker-Planck-Kolmogorov equation to derive a partial differential equation (PDE) system

$$\begin{aligned}
\partial p_{\text{hyb}}(\boldsymbol{h}_t, t|\theta)/\partial t &= -\nabla \cdot \left[\left(f_\theta(\boldsymbol{h}_t, t) + \boldsymbol{\gamma} \circ r_{\tilde{\xi}}(\boldsymbol{h}_t, t)\right) p_{\text{hyb}}(\boldsymbol{h}_t, t|\theta)\right] \\
&\quad + \nabla \cdot \left(\mathbf{1}\nabla \cdot G(\boldsymbol{h}_t, t) p_{\text{hyb}}(\boldsymbol{h}_t, t|\theta)\right), \\
\partial p_{\text{pri}}(\boldsymbol{h}_t, t)/\partial t &= -\nabla \cdot \left[\left(\boldsymbol{\gamma} \circ r_{\tilde{\xi}}(\boldsymbol{h}_t, t)\right) p_{\text{pri}}(\boldsymbol{h}_t, t)\right] \\
&\quad + \nabla \cdot \left(\mathbf{1}\nabla \cdot G(\boldsymbol{h}_t, t) p_{\text{pri}}(\boldsymbol{h}_t, t)\right),
\end{aligned}$$

where $\nabla\cdot$ is the divergence operator and $\mathbf{1} = (1, \ldots, 1)^\top$. Theoretically, these distributions can be obtained by solving the Fokker-Planck PDE. As this requires solving a PDE that is not analytically tractable, we instead resort to the discrete-time Euler-Maruyama integration.

### 6.6.2 *Proofs*

This section gives a more detailed derivation and proofs of the individual results stated in the main text.

*Lemma 1 and Proof*

LEMMA 1. *For the two process distributions*

$$\begin{aligned}
P_{0\to T}(\boldsymbol{h}_{0\to T}, \theta) &\triangleq p_{\text{pri}}(\boldsymbol{h}_{0\to T}) p_{\text{pri}}(\theta), \\
Q_{0\to T}(\boldsymbol{h}_{0\to T}, \theta) &\triangleq p_{\text{hyb}}(\boldsymbol{h}_{0\to T}|\theta) p_{\text{pri}}(\theta),
\end{aligned}$$

*as introduced in the main text the following property holds*

$$\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right) =$$

$$\frac{1}{2}\int_0^T \mathbb{E}_{Q_{0\to T}}\left[f_\theta(\boldsymbol{h}_t,t)^\top \mathbf{J}_t^{-1} f_\theta(\boldsymbol{h}_t,t)\right]\mathrm{d}t + \mathrm{KL}\left(p_\phi(\theta) \parallel p_{\mathrm{pri}}(\theta)\right),$$

*for some* $T > 0$, *where* $\mathbf{J}_t = G(\boldsymbol{h}_t,t)G(\boldsymbol{h}_t,t)^\top$ .

PROOF. Assume that the Euler-Maruyama discretization for the process $Q_{0\to T}$ on arbitrarily chosen $K$ time points within the interval $[0,T]$ has been performed. Then we have that the KL divergence between $Q$ and $P$ up to the discretization into $K$ time points $t_0,\ldots,t_K$ is given as

$$\begin{aligned}
&\mathrm{KL}\left(Q \parallel P\right) = \\
&\quad\iint \log \frac{\prod_{k=0}^{K-1}\mathcal{N}\left(\boldsymbol{h}_{k+1}|\boldsymbol{h}_k + \left(f_\theta(\boldsymbol{h}_k,t_k) + \boldsymbol{\gamma}\circ r_\xi(\boldsymbol{h}_k,t_k)\right)\Delta t_k, \mathbf{J}_k\Delta t_k\right)}{\prod_{k=0}^{K-1}\mathcal{N}\left(\boldsymbol{h}_{k+1}|\boldsymbol{h}_k + \boldsymbol{\gamma}\circ r_\xi(\boldsymbol{h}_k,t_k)\Delta t_k, \mathbf{J}_k\Delta t_k\right)} \\
&\qquad\cdot \frac{\cancel{p(\boldsymbol{h}_0)}p_\phi(\theta)}{\cancel{p(\boldsymbol{h}_0)}p_{\mathrm{pri}}(\theta)} Q_{0\to T}\,\mathrm{d}\boldsymbol{h}\,\mathrm{d}\theta \\
&= \sum_{k=0}^{K-1}\mathbb{E}_{Q_{0\to T}}\left[\log \frac{\mathcal{N}\left(\boldsymbol{h}_{k+1}|\boldsymbol{h}_k + \left(f_\theta(\boldsymbol{h}_k,t_k) + \boldsymbol{\gamma}\circ r_\xi(\boldsymbol{h}_k,t_k)\right)\Delta t_k, \mathbf{J}_k\Delta t_k\right)}{\mathcal{N}\left(\boldsymbol{h}_{k+1}|\boldsymbol{h}_k + \boldsymbol{\gamma}\circ r_\xi(\boldsymbol{h}_k,t_k)\Delta t_k, \mathbf{J}_k\Delta t_k\right)}\right] \\
&\qquad + \mathrm{KL}\left(p_\phi(\theta) \parallel p_{\mathrm{pri}}(\theta)\right),
\end{aligned}$$

where $\Delta t_k \triangleq t_{k+1} - t_k$, and $\boldsymbol{J}_k \triangleq G(\boldsymbol{h}_k,t_k)G(\boldsymbol{h}_k,t_k)^\top$. In each of the $K$ terms the expectation of the log ratio now reduces to the Kullback-Leibler divergence between two multivariate normal distributions conditioned on the other $\boldsymbol{h}_k$, which given the shared covariances and very similar means of the multivariate two normal distributions reduces to

$$\begin{aligned}
&\mathrm{KL}\left(Q \parallel P\right) = \\
&\quad\sum_{k=0}^{K-1}\mathbb{E}_{Q_{0\to T}}\left[\frac{1}{2} f_{\theta_t}(\boldsymbol{h}_k,t_k)^\top \boldsymbol{J}_k^{-1} f_{\theta_t}(\boldsymbol{h}_k,t_k)\Delta t_k\right] + \mathrm{KL}\left(p_\phi(\theta) \parallel p_{\mathrm{pri}}(\theta)\right).
\end{aligned}$$

With the limit of $K \to \infty$ we then arrive at the desired property

$$\begin{aligned}
&\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right) = \\
&\quad\frac{1}{2}\int_0^T \mathbb{E}_{Q_{0\to T}}\left[f_\theta(\boldsymbol{h}_t,t)^\top \mathbf{J}_t^{-1} f_\theta(\boldsymbol{h}_t,t)\right]\mathrm{d}t + \mathrm{KL}\left(p_\phi(\theta) \parallel p_{\mathrm{pri}}(\theta)\right),
\end{aligned}$$

which would have diverged instead if we had not assumed the shared covariance terms. □

*Theorem 1*

THEOREM 1. *Let* $p(\boldsymbol{y}_t|\boldsymbol{h}_t)$ *be a uniformly bounded likelihood function with density* $p(\boldsymbol{y}_t|\boldsymbol{h}_t)$ *everywhere and* $Q_{0\to T}$ *and* $P_{0\to T}$ *be the joints stochastic processes defined on the hypothesis class of the learning task, respectively. Define the true risk of a draw from* $Q_{0\to T}$ *on an i.i.d. sample* $\boldsymbol{Y} = \{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_K\}$ *at discrete and potentially irregular time points* $t_1,\ldots,t_K$ *drawn from an unknown ground-truth stochastic process* $\mathfrak{G}(t)$ *as the expected model misfit as on the sample as defined via the following risk over hypotheses* $H \in \mathcal{H} = (\boldsymbol{h}_{0\to T},\theta)$, i.e. a set of variables,

$$R(H) \triangleq \mathbb{E}_{\boldsymbol{Y}_k\sim\mathfrak{G}(t)}\left[1 - \frac{1}{\overline{B}_K}\prod_{k=1}^{K} p(\boldsymbol{y}_k|\boldsymbol{h}_k)\right],$$

*for time horizon $T > 0$ and the corresponding empirical risk on a data set $\mathcal{D} = \{\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_N\}$ as*

$$R_{\mathcal{D}}(H) \triangleq \frac{1}{N} \sum_{n=1}^{N} \left( 1 - \frac{1}{\overline{B}_K} \prod_{kn=1}^{K} p(\boldsymbol{y}_k^n | \boldsymbol{h}_k^n) \right),$$

where we defined

$$\overline{B}_K \triangleq \max_{\boldsymbol{y}_k, \boldsymbol{h}_k} \prod_{k=1}^{K} p\left(\boldsymbol{y}_k | \boldsymbol{h}_k\right) \leq \left( \max_{\boldsymbol{y}_k, \boldsymbol{h}_k} p\left(\boldsymbol{y}_k | \boldsymbol{h}_k\right) \right)^K.$$

*Then the expected true risk is bounded above with probability $\mathbb{P} \geq 1 - \delta$ for some $\delta \in [0, 1]$by the marginal negative log-likelihood of the predictor and a complexity functional as*

$$\begin{aligned}
\mathbb{E}_{H \sim Q_{0\to T}} \left[ R(H) \right] &\leq \mathbb{E}_{H \sim Q_{0\to T}} \left[ R_{\mathcal{D}}(H) \right] + \mathcal{C}_\delta(Q_{0\to T}, P_{0\to T}) \qquad (6.17) \\
&\leq -\frac{1}{N} \sum_{n=1}^{N} \log \left( \frac{1}{S} \sum_{s=1}^{S} \prod_{k=1}^{K} p(\boldsymbol{y}_k^n | \boldsymbol{h}_k^{n,s}) \right) + \mathcal{C}_{\delta/2}(Q_{0\to T}, P_{0\to T}) \\
&\quad + \sqrt{\frac{\log(2N/\delta)}{2S}} + K \log \overline{B} \\
&\leq -\frac{1}{SN} \sum_{n=1}^{N} \sum_{s=1}^{S} \sum_{k=1}^{K} \log \left( p(\boldsymbol{y}_k^n | \boldsymbol{h}_k^{s,n}) \right) + \mathcal{C}_{\delta/2}(Q_{0\to T}, P_{0\to T}) \\
&\quad + \sqrt{\frac{\log(2N/\delta)}{2S}} + K \log \overline{B}, \qquad (6.18)
\end{aligned}$$

*where $S$ is the sample count taken independently for each observed sequence, and the complexity functional is given as*

$$\mathcal{C}_\delta(Q_{0\to T}, P_{0\to T}) \triangleq \sqrt{\frac{\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right) + \log(4\sqrt{N}/\delta)}{2N}}$$

*with $\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right)$ as in Lemma 1.*

*Proof of Theorem 1*

PROOF. To be able to apply known PAC-Bayes bounds, we first define the hypothesis class $H \in \mathcal{H}_K$ that contains the latent states $\boldsymbol{h}_k, \theta$ that explain the observations $\boldsymbol{y}_k$. The data set $\mathcal{D} = \{\boldsymbol{Y}_k^n\}_{k,n}$ was generated by an unknown stochastic process $\mathfrak{G}(t)$.

Note that we normalize the risks $R(H)$ and $R_{\mathcal{D}}(H)$ by the maximum of the likelihood and thereby obtaining a possible range of these risk of $[0, 1]$. This bounding is achievable, as we model the term $p(\boldsymbol{y}_k | \boldsymbol{h}_k)$, by a normal distribution with a fixed variance, is upper bounded by its mode.

The rescaling to $[0, 1]$ allows us to rely on the binary Kullback-Leibler divergence,

$$\mathrm{kl}\left(q \parallel p\right) \triangleq q \log \frac{q}{p} + (1 - q) \log \frac{1 - q}{1 - p},$$

to measure the distance between the empirical and the true risk getting the following bound due to Maurer (2004)

$$\mathrm{kl}\left(\mathbb{E}_H\left[R_{\mathcal{D}}(H)\right] \parallel \mathbb{E}_H\left[R(H)\right]\right) \leq \frac{\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{N},$$

which wholds for $N > 8$, and $\delta \in (0,1)$ with probability $\mathbb{P} \geq 1-\delta$.

To optimally use it as a generalization bound, we require an inverse. In that case, a numerical inverse is a suitable approach to ensure tightness. To get an analytically tractable bound, we instead follow common practice and use Pinsker's inequality (Tolstikhin and Seldin, 2013; Dziugaite and Roy, 2017),

$$|p-q| \leq \sqrt{\mathrm{kl}\left(q \parallel p\right)/2}.$$

This allows us to compute the following tractable bound.

PAC-BAYES BOUND *For any $[0,1]$-valued loss function giving rise to empirical and true risk $R_{\mathcal{D}}(H), R(H)$, for any distribution $\Delta$, for any $N > 8$, for any distribution $P_{0\to T}$ on a hypothesis set $\mathcal{H}_K$, and for any $\delta \in (0,1)$, the following holds with probability at least $1-\delta$ over the training set $\mathcal{D} \sim \Delta^N$:* *PAC-Bayes Bound*

$$\forall Q_{0\to T}: \quad \mathbb{E}_{H\sim Q_{0\to T}}\left[R(H)\right] \leq \mathbb{E}_{H\sim Q_{0\to T}}\left[R_{\mathcal{D}}(H)\right] + \sqrt{\frac{\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}$$

Here, $\mathrm{KL}\left(Q_{0\to T} \parallel P_{0\to T}\right)$ acts as a complexity measure that measures how much the posterior predictive governing the SDE $Q_{0\to T}$ needed to be adapted to the data when compared to an a priori chosen SDE that could alternatively have generated data $P_{0\to T}$. In our situation, $Q_{0\to T}$ is obtained by our approximation scheme, resulting in a bounded likelihood of observations $\boldsymbol{y}_k$ which factorizes over different observations $n$. The $P_{0\to T}$ can be arbitrarily chosen as long as it does not depend on the observations. As mentioned in the main discussion, we chose an SDE with the same diffusion term, which also factorizes over observations. Using this setting, we can analytically compute the KL-distance (as shown in Lemma 1).

On the right-hand side of the boundbound, we need to evaluate $\mathbb{E}_{H\sim Q_{0\to T}}\left[R_{\mathcal{D}}(H)\right]$ which we can approximate as

$$\begin{aligned}
\mathbb{E}_{H\sim Q_{0\to T}}\left[R_{\mathcal{D}}(H)\right] &= \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{H\sim Q_{0\to T}}\left[1 - \frac{1}{\overline{B}_K}\prod_{k=1}^{K} p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^n)\right] \\
&= 1 - \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{H\sim Q_{0\to T}}\left[\frac{1}{\overline{B}_K}\prod_{k=1}^{K} p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^n)\right] \\
&\leq 1 - \frac{1}{SN}\sum_{n=1}^{N}\sum_{s=1}^{S}\left[\frac{1}{\overline{B}_K}\prod_{k=1}^{K} p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^{n,s})\right]
\end{aligned}$$

$$
\begin{aligned}
&+\sqrt{\frac{\log(2N/\delta)}{2S}} \qquad \textit{(via Hoeffding's inequality)} \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(1-\frac{1}{S}\sum_{s=1}^{S}\frac{1}{\overline{B}_K}\prod_{k=1}^{K}p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^{n,s})\right) \\
&+\sqrt{\frac{\log(2N/\delta)}{2S}} \\
&\leq -\frac{1}{N}\sum_{n=1}^{N}\log\left(\frac{1}{S}\sum_{s=1}^{S}\prod_{k=1}^{K}p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^{n,s})\right) \\
&+\log\overline{B}_K+\sqrt{\frac{\log(2N/\delta)}{2S}} \qquad \textit{(as } -\log z\geq 1-z\textit{)} \\
&\leq -\frac{1}{SN}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{k=1}^{K}\left[\log p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^{n,s})\right] \\
&+\log\overline{B}_K+\sqrt{\frac{\log(2N/\delta)}{2S}} \quad \textit{(via Jensen's inequality),}
\end{aligned}
$$

where we have used Hoeffding's inequality for estimating the true expectation over hypotheses with a $K$ samples trace $\boldsymbol{h}_k^{n,s}$ with $k=1,\ldots,K, s=1,\ldots,S$ for each observation $n$. As we approximate the integral for each time-series $n$ separately via sampling, we require Hoeffding to hold simultaneously for all $n$. Using a union bound, we have to scale $\delta$ for each $n$ by $N$. Splitting confidences between the PAC-Bayes bound and the sampling based approximation results an additional factor of 2. With $\delta/(2N)$, the corresponding inequality holds with a probability of $\mathbb{P}>\delta/2$. Also using $\delta/2$ in PAC-theorem, we obtain that with $\mathbb{P}\geq 1-\delta$ we have for all $Q_{0\to T}$ that

$$
\begin{aligned}
&\mathbb{E}_{H\sim Q_{0\to T}}\left[R(H)\right] \\
&\leq \mathbb{E}_{H\sim Q_{0\to T}}\left[R_{\mathcal{D}}(H)\right]+\sqrt{\frac{\mathrm{KL}\left(Q_{0\to T}\parallel P_{0\to T}\right)+\log\left(\frac{2\sqrt{N}}{\delta/2}\right)}{2N}} \\
&\leq -\frac{1}{N}\sum_{n=1}^{N}\log\left(\frac{1}{S}\sum_{s=1}^{S}\prod_{k=1}^{K}p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^{n,s})\right)+\log\overline{B}_K+\sqrt{\frac{\log(2N/\delta)}{2S}} \\
&\quad+\sqrt{\frac{\mathrm{KL}\left(Q_{0\to T}\parallel P_{0\to T}\right)+\log\left(\frac{4\sqrt{N}}{\delta}\right)}{2N}} \\
&\leq -\frac{1}{SN}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{k=1}^{K}\log p(\boldsymbol{y}_k^n|\boldsymbol{h}_k^{n,s})+\log\overline{B}_K+\sqrt{\frac{\log(2N/\delta)}{2S}} \\
&\quad+\sqrt{\frac{\mathrm{KL}\left(Q_{0\to T}\parallel P_{0\to T}\right)+\log\left(\frac{4\sqrt{N}}{\delta}\right)}{2N}}
\end{aligned}
$$

□

*Corollary 1* COROLLARY 1. *Given a L-Lipschitz continuous function set*

$$
\left\{f_\theta^n(x):\mathbb{R}\to[0,1]\middle| n=1,\cdots,N\right\}\bigcup\left\{g_\theta(x):\mathbb{R}\to[0,+\infty]\right\},
$$

*for the two losses:*

$$l_1(\theta) = -\sum_{n=1}^{N} f_\theta^n(x) + g_\theta(x) \;\; \textit{and} \;\; l_2(\theta) = -\sum_{n=1}^{N} \log f_\theta^n(x) + g_\theta(x),$$

*the sequential updates (with $\theta^0 \triangleq \theta$)*

$$\begin{aligned} \theta^{(n)} &\leftarrow \theta^{(n-1)} + \alpha_n \nabla \big( \log f^n_{\theta^{(n-1)}}(x) \big), \; n = 1, \dots, N, \\ \theta^{(N+1)} &\leftarrow \theta^{(N)} - \alpha_{N+1} \nabla g_{\theta^{(N)}}(x), \end{aligned}$$

*where $\alpha_n \in (0, f^n_{\theta^{(n-1)}}(x)/L)$ $\forall n$ and $\alpha_{N+1} \in (0, 1/L)$, satisfy both $l_1(\theta^{(N+1)}) \leq l_1(\theta)$ and $l_2(\theta^{(N+1)}) \leq l_2(\theta)$.*

*Proof of Corollary 1*

PROOF. As we only consider updates in $\theta$ for constant $x$, we simplify the notation for this proof to $f^n(\theta) := f^n_\theta(x)$, $g(\theta) = g_\theta(x)$. I.e. we have as the two loss terms

$$l_1(\theta) = -\sum_{n=1}^{N} f^n(\theta) + g(\theta) \quad \text{and} \quad l_2(\theta) = -\sum_{n=1}^{N} \log f^n(\theta) + g(\theta).$$

In general we have with $\log f(\theta) < f(\theta)$ that $l_1(\theta) < l_2(\theta)$. Similarly we have

$$\nabla l_2(\theta) = -\sum_n \underbrace{\frac{1}{f^n(\theta)}}_{\geq 1} \nabla f^n(\theta) + \nabla g(\theta) \leq -\sum_n \nabla f^n(\theta) + \nabla g(\theta) = \nabla l_1(\theta).$$

Due to the sequential updates we can consider each term separately. For an $L$-Lipschitz function $f^n(\theta)$, we have that for arbitrary $x, y$

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2} ||y - x||_2^2.$$

Choosing $y = \theta^{(n-1)}$ and $x = \theta^{(n)} = \theta^{(n-1)} + \alpha_n \nabla \log f^n$ this gives us

$$\begin{aligned} f(\theta^{(n-1)}) &\leq f(\theta^{(n)}) - \frac{\alpha_n}{f^n(\theta^{(n)})} ||\nabla f^n(\theta^{(n)})||_2^2 + \frac{L\alpha_n^2}{2 f^n(\theta^{(n)})^2} ||\nabla f^n(\theta^{(n)})||_2^2 \\ &= f^n(\theta^{(n)}) - \underbrace{\frac{\alpha_n}{f^n(\theta^{(n)})}}_{\geq 0} \underbrace{\left(1 - \frac{L\alpha_n}{2 f^n(\theta^{(n)})}\right)}_{> 0} ||\nabla f^n(\theta^{(n)})||_2^2 \\ &\leq f^n(\theta^{(n)}), \end{aligned}$$

and hence chaining the update steps gives the desired result. □

That is, updating the terms in $l_2(\theta)$ sequentially, one can ensure concurrent optimization of $l_1(\theta)$. Note that $l_1(\theta)$ and $l_2(\theta)$ are not necessarily dual objectives, hence may have different extrema. Nevertheless, a gradient step that decreases one loss also decreases the other with potentially a different magnitude. In practice, we observe this behaviour to also hold empirically for joint gradient update steps with shared learning rates. Applying Lemma 2 to Theorem 1, we establish a useful link between empirical Bayes and PAC-Bayes learning.

*Theorem 2*

THEOREM 2 (STRONG CONVERGENCE). *Let $\boldsymbol{h}_t^\theta$ be an Itô process as in* (6.3) *with drift parameters $\theta$ and its Euler-Maruyama approximation $\widetilde{\boldsymbol{h}}_t^\theta$ for some regular step size $\Delta t > 0$. For some coefficient $R > 0$ and any $T > 0$, the following inequality holds as $S \to \infty$*

$$\mathbb{E}\left[\left|\mathbb{E}_\theta[\boldsymbol{h}_T^\theta] - \frac{1}{S}\sum_{s=1}^{S}\widetilde{\boldsymbol{h}}_T^{\theta^{(s)}}\right|\right] \leq R\Delta t^{1/2},$$

*where $\{\theta^{(s)} \sim p_\phi(\theta) | s = 1, \ldots, S\}$ are i.i.d. draws from a prior $p_\phi(\theta)$.*

*Proof of Theorem 2*

PROOF. The Euler-Maruyama approximation converges strongly as

$$\mathbb{E}\left[\left|\boldsymbol{h}_T^\theta - \widetilde{\boldsymbol{h}}_T^\theta\right|\right] \leq R\Delta t^{1/2},$$

for a positive constant $R$ and a suitably small step size $\Delta t$ as discussed for example by Kloeden and Platen (2011). To simplify the mathematical notation we follow their approach of comparing the absolute error of the end of the trajectory throughout the proof. As our sampling scheme is unbiased it is a consistent estimator and we have that asymptotically for $S \to \infty$

$$\frac{1}{S}\sum_{s=1}^{S}\widetilde{h}_T^{\theta(s)} = \mathbb{E}_\theta\left[\widetilde{h}_T^\theta\right].$$

We then have for the marginal $\boldsymbol{h}_T$, $\tilde{\boldsymbol{h}}_T$ that

$$\begin{aligned}
\mathbb{E}\left[\left|\boldsymbol{h}_T - \widetilde{\boldsymbol{h}}_T\right|\right] &= \mathbb{E}\left[\left|\mathbb{E}_\theta\left[\boldsymbol{h}_T^\theta\right] - \mathbb{E}_\theta\left[\widetilde{\boldsymbol{h}}_T^\theta\right]\right|\right] \\
&= \mathbb{E}\left[\left|\mathbb{E}_\theta\left[\boldsymbol{h}_T^\theta - \widetilde{\boldsymbol{h}}_T^\theta\right]\right|\right] \\
&\leq \mathbb{E}\left[\mathbb{E}_\theta\left[\left|\boldsymbol{h}_T^\theta - \widetilde{\boldsymbol{h}}_T^\theta\right|\right]\right] \\
&\leq \mathbb{E}_\theta\left[R\Delta t^{1/2}\right] = R\Delta t^{1/2},
\end{aligned}$$

where the first inequality is due to Jensen and the second due to the strong convergence result for a fixed set of parameters after switching the order of the two expectation. □

### 6.6.3 *Further Details On The Experiments*

Here we provide further details on the experimental setup we used in obtaining our results reported in the main discussion. We observed our results to be robust against most of the design choices.

LOTKA VOLTERRA. We took $10^5$ Euler-Maruyama steps on the interval $[0, 10]$ with a time step size of $10^{-4}$, downsampling them by a factor of 100 giving us 1000 observations with a frequency of 0.01. We take the first 500 observations on the interval [0,5] to be the training data and the observations in $(5, 10]$ to be the test data. Each sequence is split into ten sequences of length

50. Assuming the diffusion parameters to be known and fixed, both BNSDEs (i.e. with and without prior knowledge) get a four-layer net as the drift function with 50 neurons per layer and ReLU activation functions. The BNSDE with prior knowledge as well as the raw SDE estimate each gets an initial sample of $\tilde{\boldsymbol{\theta}}$ parameters as the prior information by sampling from a normal distribution centered around the true parameters, i.e. $\tilde{\boldsymbol{\theta}} \sim \mathcal{N}(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}, \sigma^2 \mathbb{1}_4)$. The models are each trained for 50 epochs with the Adam optimizer (Kingma and Ba, 2015) and a learning rate of 0.001. Since both the latent and observed spaces are only two dimensional, we did not need an observation model in this experiment. We directly linked the BNSDE to the likelihood.

LORENZ ATTRACTOR. We took 200.000 Euler-Maruyama steps ahead with a time step size of $10^{-4}$ and downsampling by a factor 0.01, which gives a sequence of 2000 observations with frequency 0.01. We split the first half of this data set into 20 sequences of length 50 and use them for training, and the second half to 10 sequences of length 100 and use for testing. For all model variants, we used an Adam optimizer learning rate 0.001, minibatch size of two, a drift net with two hidden layers of 100 neurons and softplus activation function.We trained all models for 100 epochs and observed this training period to be sufficient for convergence.

CMU MOTION CAPTURE. In this experiment, we tightly follow the design choices reported by Yildiz et al. (2019) to maintain comparability. This setup assumes the stochastic dynamics are determined in a six-dimensional latent space. Yildiz et al. (2019) use an auto-encoder to map this latent space to the 50-dimensional observation space back and forth. We adopt their exact encoder-decoder architecture and incorporate it into our BNSDE, arriving at the data generating process

$$
\begin{aligned}
\theta_f &\sim p_{\phi_f}(\theta_f), \\
d\boldsymbol{h}_t|\theta_f &\sim f_{\theta_f}\big(b_\lambda(\boldsymbol{h}_t), t\big)dt + G\big(b_\lambda(\boldsymbol{h}_t), t\big)d\boldsymbol{\beta}_t, \\
\boldsymbol{z}_t|\boldsymbol{h}_t &\sim \mathcal{N}(\boldsymbol{z}_t|a_\psi(\boldsymbol{h}_t), 0.5 \cdot 10^{-6}\mathbb{1}), \\
\boldsymbol{y}_t|\boldsymbol{z}_t &\sim \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{z}_t, 0.5 \cdot 10^{-6}\mathbb{1}), \qquad \forall t \in \boldsymbol{t}.
\end{aligned}
$$

Above, $b_\lambda(\cdot,\cdot)$ is the encoder that takes the observations of the last three time points as input, passes them through two dense layers with 30 neurons and softplus activation function, and then linearly projects them to a six-dimensional latent space, where the dynamics are modeled. The decoder $a_\psi(\boldsymbol{h}_t)$ follows the same chain of mapping operations in reverse order. The only difference is that the decoder's output layer emits only one observation point, as opposed to the encoder admitting three points at once.

The drift function $f_{\theta_f}(\cdot,\cdot)$ is governed by another separate Bayesian neural net, again with one hidden layer of 30 neurons and softplus activation function on the hidden layer. The diffusion function is fixed to be constant.

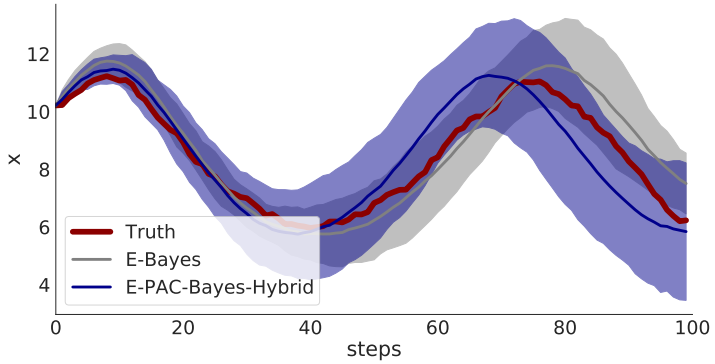
We train all models except SGLD with the Adam optimizer for 3000 epochs on seven randomly chosen snippets at a time with a learning rate of $10^{-3}$. We

use snippet length 30 for the first 1000 epochs, 50 until epoch 2500, and 100 afterwards. SGLD demonstrates significant training instability for this learning rate, hence for it we drop its learning rate to the largest possible, stable value $10^{-5}$ and increase the epoch count to 5000.

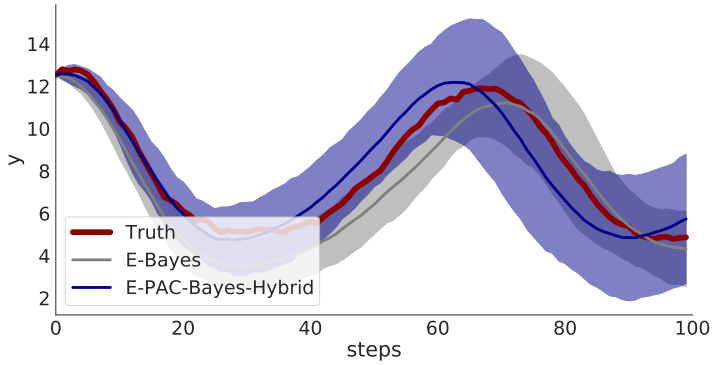
### 6.6.4 *Further Results on the Lorenz Attractor*

As discussed above, the model is trained solely on the first 1000 observations of a trajectory consisting of 2000 observations, leaving the second half for the test evaluation. Figure 6.4 visualizes the qualitative difference between the two. Note also the single loop the trajectory performs, which we will see again in the 1d projections below. To visualize explore the qualitative difference of our proposed model with weak prior knowledge compared to one lacking this knowledge, we consider the situation where we have structural prior knowledge only about the third SDE (i.e. the penultimate case in Table 1 with $\gamma = [0, 0, 1]$.

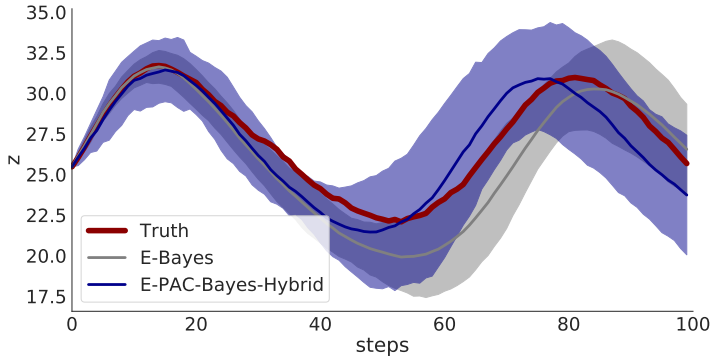
To properly visualize it, we switch from the 3d plot to 1d plots, always showing one of the three dimensions vs the time component. We always start at $T = 10$, forecasting either 100 steps (as in the numerical evaluation), 200 or 1000 steps. All the following figures show the mean trajectory averaged over 21 trajectories, as well as an envelope of $\pm$ two standard deviations. Figure 6.6 visualizes that the qualitative behaviour is similar without clear differences at that time scale. Doubling the predicted time interval as shown in Figure 6.7 the baseline starts to diverge from the true test sequence, while our proposed model still tracks it closely, be it at an increased variance. Finally, predicting for 1000 time steps (Figure 6.8) the Lorenz attractor's chaotic behaviour becomes visible as the mean in both setups no longer tracks the true trajectory. However, the baseline keeps a rather small variance and a strong tendency in its predictions that do not replicate the Lorenz attractor's qualitative behaviour. While the proposed model also shows an unreliable average, the large variance, which nearly always includes the true trajectory, shows that the qualitative behaviour is still replicated properly by individual trajectories of the model. See Figure 6.9 for seven individual trajectories of each of the two models. All trajectories of *E-PAC-Bayes-Hybrid* show the qualitatively correct behaviour, including even the characteristic loop.




**(a)** $x$ coordinate over time

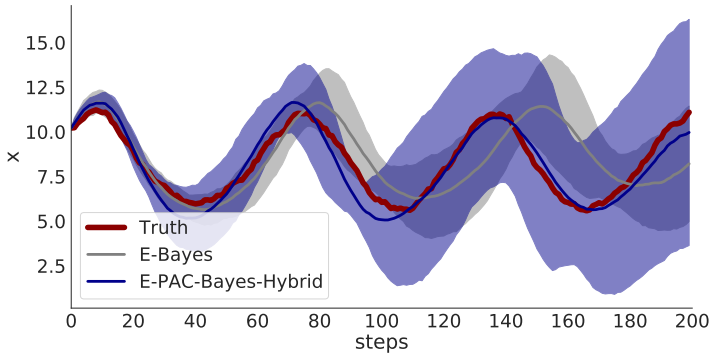



**(b)** $y$ coordinate over time

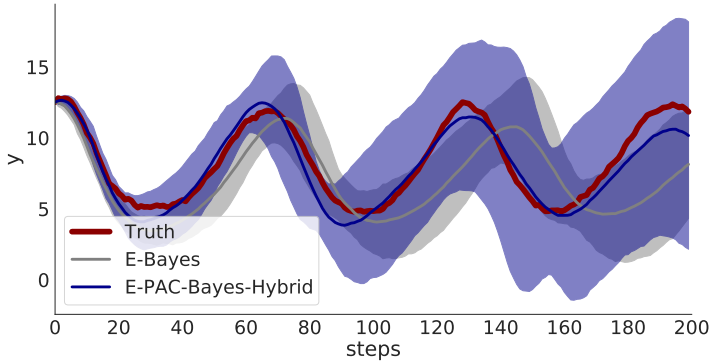



**(c)** $z$ coordinate over time

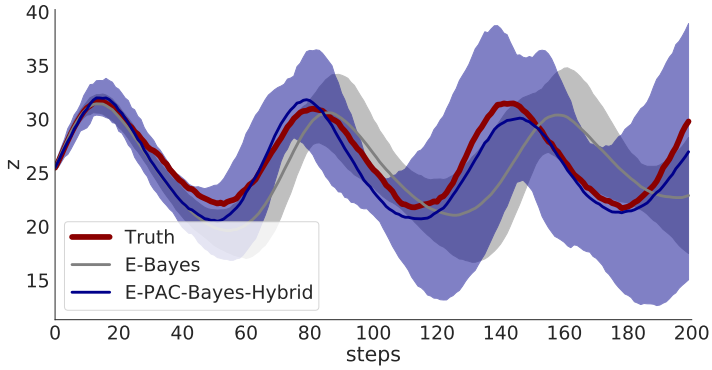
**Figure 6.6:** Predicting 100 time steps ahead. The hybrid model has prior knowledge of $\mathrm{d}z$. The bold lines give the mean over 21 trajectories the shading specifies $\pm$ two standard deviations.




**(a)** $x$ coordinate over time

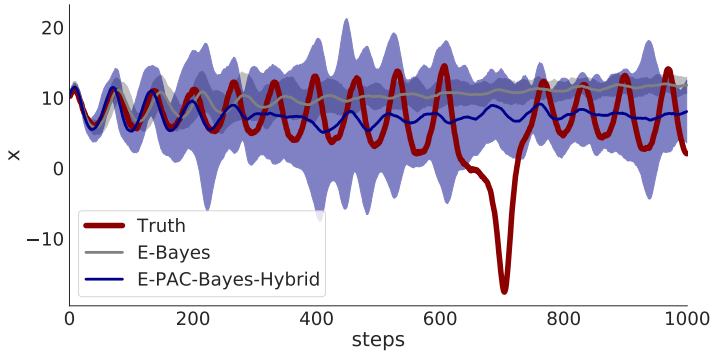



**(b)** $y$ coordinate over time

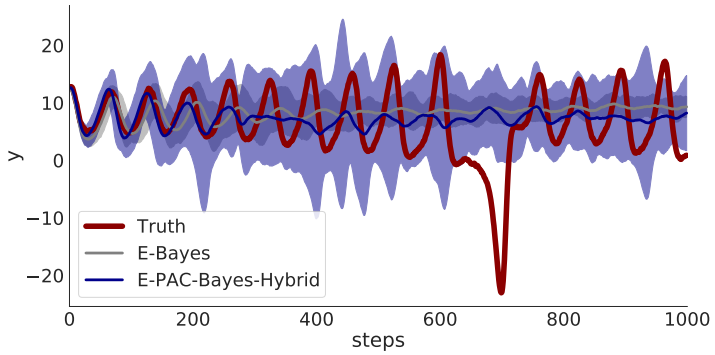



**(c)** $z$ coordinate over time

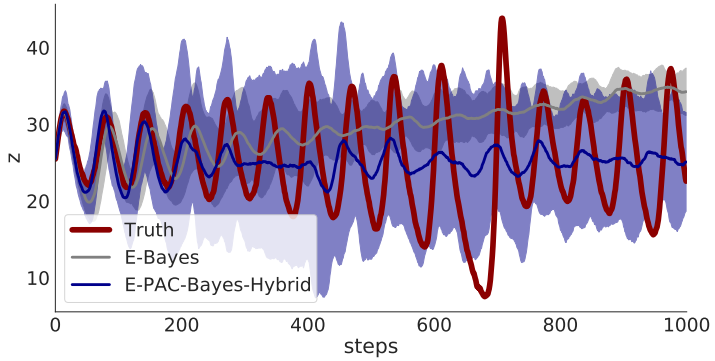
**Figure 6.7:** Predicting 200 time steps ahead. The hybrid model has prior knowledge of $\mathrm{d}z$. The bold lines give the mean over 21 trajectories the shading specifies $\pm$ two standard deviations.




**(a)** $x$ coordinate over time

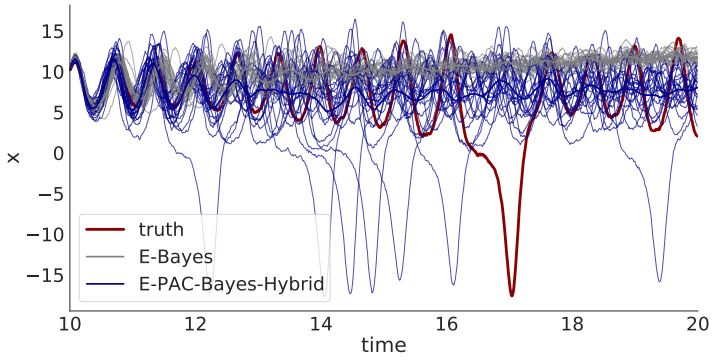



**(b)** $y$ coordinate over time

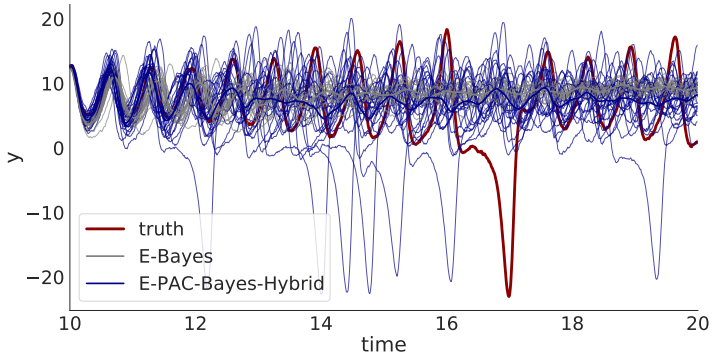



**(c)** $z$ coordinate over time

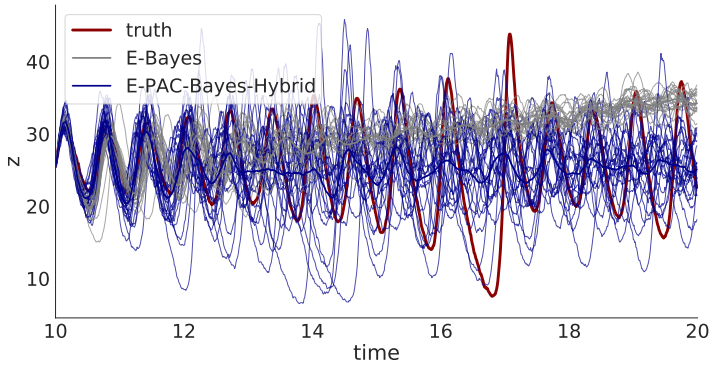
**Figure 6.8:** Predicting 1000 time steps ahead. The hybrid model has prior knowledge of $\mathrm{d}z$. The bold lines give the mean over 21 trajectories the shading specifies $\pm$ two standard deviations.




**(a)** $x$ coordinate over time




**(b)** $y$ coordinate over time




**(c)** $z$ coordinate over time

**Figure 6.9:** Predicting 1000 time steps ahead. The hybrid model has prior knowledge of $\mathrm{d}z$. The bold lines give the mean over 21 trajectories the shading specifies $\pm$ two standard deviations. Each line shows one sampled trajectory.

# 7

# CONCLUSION

As set out in the introduction, this thesis aimed to provide evidence for the usefulness of Bayesian neural nets and to introduce new approaches for learning such probabilistic models.

Training BNNs via variational inference already greatly improves their scalability. However, common approaches still rely on samples of the weight parameters during the training procedure leading to either noisy gradients when relying on few samples or expensive ones if multiple forward passes through the layers are combined to stabilize them. We introduced a new approach that relies on the structure of piecewise linear activation functions to train them more stably and demonstrated their usefulness empirically.

We also showed how to use central limited theorem based moment matching approaches for active learning, introducing a new approach to learn the acquisition function via a second Bayesian neural net in parallel to the main learner, instead of having to rely on hard-coded approaches. Additionally, we demonstrated how to adapt the concept of type-II Maximum Likelihood to the learning of BNNs and how to modify PAC-Bayes generalization bounds as regularizing objectives.

We used these approaches to adapt the usually deterministic methods for out-of-distribution detection via evidential deep learning and the modeling of stochastic dynamics via neural stochastic differential equation models to be probabilistic via BNNs.

To summarize, we can say that Bayesian neural nets indeed have their legitimacy and practical usefulness. However, looking ahead, we have to acknowledge that this research direction still poses a lot of open question, and the field is far from having converged. All of the BNNs we discussed relied either on the simplifying assumption of mean-field normal posteriors or factorized normal priors in the case of the empirical Bayes approach. While it is a huge step ahead from being restricted to pure MCMC methods, it is also clear that this is a restriction purely motivated by theoretical constraints and is a poor approximation to the true desired posterior, which is highly multimodal. And even these simplified mean-field approaches struggle to compete with the really deep neural nets employed in computer vision and natural language processing tasks.

And yet, this is not to be read as a discouraging summary. This thesis relied on comparatively smallish nets throughout the experiments to allow for proper evaluations and comparisons to the baselines. Also, it relied on the pure approach of training fully probabilistic nets from scratch. Removing these

restrictions in practical applications, there has been a lot of progress relying for example on deterministic pretraining, or mixtures of deterministic and probabilistic layers to improve the scalability, which tend to consistently show that the extra cost of probabilistic layers is justified by improved calibration of the predictive uncertainties etc. . Additionally, every recent conference offers new ideas of how to improve the variational posterior, rely on ensembling strategies to better explore the posterior landscape, or make use of flow-based approaches for increased flexibility. While we are probably far from the global optimum still, the field seems definitely not stuck in a local optimum and progresses with strong gradients.

Instead of further torturing the metaphor, we thank the reader for having accompanied us on the journey through this thesis and which him/her all the best. May we always be able to say: *"Servir la science, c'est notre joie!"*[1]

1 Motto of Podcast Science (`https://www.podcastscience.fm/`).

# 8

# APPENDIX

This chapter collects some notation and definitions of the different distributions and inequalities used throughout the thesis.

RELATIONSHIP NOTATION. In addition to the common equality and inequality relations, we use the following five relations in the equations throughout this thesis, where *lhs* refers to the left-hand side of the equation and *rhs* to the right-hand side.

| | |
|---|---|
| $\triangleq$ | lhs is equal to rhs by definition |
| $\overset{c}{=}$ | lhs is equal to rhs up to an additive constant |
| $\propto$ | lhs is equal to rhs up to a multiplicative constant (i.e. proportional) |
| $\approx$ | lsh is approximately equal to rhs |
| $\leftarrow$ | lhs gets assigned the value on the rhs |

## 8.1 DISTRIBUTIONS

We rely on various distributions in this thesis, whose density and parametrization are summarized in this section.

NORMAL/GAUSSIAN *The normal distribution for $x \in \mathbb{R}$ is parametrized by a mean $\mu \in \mathbb{R}$, and either a variance $\sigma^2 \in \mathbb{R}_+$, or a precision $\beta = \frac{1}{\sigma^2}$. Its density is given as* *Normal/Gaussian*

$$\mathcal{N}(x|\mu,\sigma^2) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(x-\mu)^2\right)$$

Its generalization to a $d$-dimensional $\boldsymbol{x}$ is given as

MULTIVARIATE NORMAL *The multivarate normal distribution for $\boldsymbol{x} \in \mathbb{R}^d$ is parameterized by a mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$, and a covariance matrix $\boldsymbol{\Sigma}$, which is constrained to be symmetric positive definite. Its density is given via* *Multivariate Normal*

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

*where $|\boldsymbol{\Sigma}|$ refers to the determinant. The Kullback-Leibler divergence between two multivarate normal distributions is given as*

$$\mathrm{KL}\left(\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}) \parallel \mathcal{N}(\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Sigma}})\right) = \frac{1}{2}\left(\mathrm{tr}\left(\tilde{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}\right) + (\tilde{\boldsymbol{\mu}}-\boldsymbol{\mu})^\top \tilde{\boldsymbol{\Sigma}}^{-1}(\tilde{\boldsymbol{\mu}}-\boldsymbol{\mu}) - d + \log\frac{|\tilde{\boldsymbol{\Sigma}}|}{|\boldsymbol{\Sigma}|}\right),$$

*where $tr(\cdot)$ is the trace operator.*

*Bernoulli*

BERNOULLI *The Bernoulli distribution for $x \in \{0,1\}$ is parameterized by $p \in [0,1]$ with a density*

$$\mathcal{B}er(x|p) \triangleq p^x(1-p)^{(1-x)}.$$

Its $C$ class generalization is given as

*Categorical*

CATEGORICAL *The Categorical distribution for $\boldsymbol{x} \in \{0,1\}^C$, $\sum_c x_c = 1$ is parameterized by $\boldsymbol{p} \in [0,1]^C$ such that $\sum_c p_c = 1$, with a density*

$$\mathcal{C}at(\boldsymbol{x}|\boldsymbol{p}) \triangleq \prod_{c=1}^{C} p_c^{x_c},$$

*Dirichlet*

DIRICHLET *The Dirichlet distribution for $\boldsymbol{x} \in [0,1]^C$, $\sum_c x_c = 1$ is parameterized by $\boldsymbol{\alpha} \in \mathbb{R}_+^C$, with a density*

$$\mathcal{D}ir(\boldsymbol{x}|\boldsymbol{\alpha}) \triangleq \frac{\Gamma(\alpha_0)}{\prod_c \Gamma(\alpha_c)} \prod_c x_c^{\alpha_c - 1},$$

*where $\Gamma(\cdot) \triangleq \int_0^\infty u^{x-1} \exp(-u)\, \mathrm{d}u$.*

## 8.2 FUNCTIONS

*Logistic sigmoid*

LOGISTIC SIGMOID *The logistic sigmoid $\sigma(\cdot) : \mathbb{R} \to [0,1]$ is defined as*

$$\sigma(a) \triangleq \frac{\exp(a)}{1+\exp(a)} = \frac{1}{1+\exp(-a)}.$$

Its generalization to $C$ dimensional input is given via the softmax.

*Softmax*

SOFTMAX *The softmax $\zeta(\cdot) : \mathbb{R}^C \to [0,1]^C$ is defined as*

$$\zeta(\boldsymbol{a})_c \triangleq \frac{\exp(\boldsymbol{a}_c)}{\sum_{c=1}^{C} \exp(\boldsymbol{a}_c)}.$$

*It has the property that $\sum_c \zeta(\boldsymbol{a})_c = 1$, i.e. it maps to the probability simplex.*

*Normal CDF/PDF*

NORMAL CDF/PDF *Throughout the thesis we refer to the standard normal probability density function as*

$$\phi(x) \triangleq \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right),$$

*and the cumulative density function of the standard normal as*

$$\Phi(x) \triangleq \int_{-\infty}^{x} \phi(t)\, \mathrm{d}t.$$

## 8.3 INEQUALITIES

See Wasserman (2013) for proofs of the following two inequalities.

JENSEN'S INEQUALITY *If $f(\cdot)$ is a convex function then* *Jensen's inequality*

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]).$$

*If $f(\cdot)$ is concave then*

$$\mathbb{E}[f(X)] \leq f(\mathbb{E}[X]).$$

HOEFFDING'S INEQUALITY *Let $X_1, \ldots, X_N$ be independent random variables with $\mathbb{E}[X_n] = 0$ and $a_n \leq X_n \leq b_n$. Let $\varepsilon > 0$, then for any $t > 0$ the following inequality holds* *Hoeffding's Inequality*

$$\mathbb{P}\left(\sum_{n=1}^{N} X_n \geq \varepsilon\right) \leq \exp(-t\varepsilon) \prod_{n=1}^{N} \exp\left(t^2(b_n - a_n)^2/8\right).$$

# BIBLIOGRAPHY


Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*.

Agnihotri, Apoorv and Nipun Batra (2020). "Exploring Bayesian Optimization". In: *Distill*.

Alquier, Pierre, James Ridgway, and Nicolas Chopin (2016). "On the Properties of Variational Approximations of Gibbs Posteriors". In: *Journal of Machine Learning Research*.

Archambeau, Cédric, Manfred Opper, Yuan Shen, Dan Cornford, and John Shawe-taylor (2008). In: *Advances in Neural Information Processing Systems*.

Ba, Jimmy and Brendan Frey (2013). "Adaptive dropout for training deep neural networks". In: *Advances in Neural Information Processing Systems*.

Baram, Yoram, Ran El-Yaniv, and Kobi Luz (2004). "Online Choice of Active Learning Algorithms". In: *Journal of Machine Learning Research*.

Barber, David (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

Bauer, Matthias, Mark v.d. Wilk, and Carl E. Rasmussen (2016). "Understanding Probabilistic Sparse GaussianProcess Approximations". In: *Advances in Neural Information Processing Systems*.

Betancourt, Michael (2017). "A Conceptual Introduction to Hamiltonian Monte Carlo". In: *arXiv preprint arXiv:1701.02434*.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. springer.

Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe (2017). "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association*.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight Uncertainty in Neural Network". In: *Proceedings of the 32nd International Conference on Machine Learning*.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). "Language Models are Few-Shot Learners". In: *arXiv preprint arXiv:2005.14165*.

Brunton, Steven L., Joshua L. Proctor, and J. Nathan Kutz (2016). "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems". In: *Proceedings of the National Academy of Sciences*.

Bui, Thang, Daniel Hernandez-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner (2016). "Deep Gaussian Processes for Regression using Approximate Expectation Propagation". In: *Proceedings of The 33rd International Conference on Machine Learning*.

Callaway, Frederick, Sayan Gul, Paul M. Krueger, Thomas L. Griffiths, and Falk Lieder (2018). "Learning to Select Computations". In: *Uncertainty in Artificial Intelligence*.

Catoni, Olivier (2007). "PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning". In: *IMS Lecture Notes Monograph Series*.

Chen, Ricky T. Q., Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud (2018a). "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*.

Chen, Tianqi, Emily Fox, and Carlos Guestrin (2014). "Stochastic Gradient Hamiltonian Monte Carlo". In: *Proceedings of the 31st International Conference on Machine Learning*.

Chen, Yutian, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas (2018b). "Bayesian Optimization in AlphaGo". In: *arXiv preprint arXiv:1812.06855*.

Chu, Hong-Min and Hsuan-Tien Lin (2016). "Can Active Learning Experience Be Transferred?" In: *International Conference on Data Mining*.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2016). "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *International Conference on Learning Representations*.

Damianou, Andreas and Neil D. Lawrence (2013). "Deep Gaussian Processes". In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*.

Deisenroth, Marc, Dieter Fox, and Carl E. Rasmussen (2015). "Gaussian Processes for Data-Efficient Learning in Robotics and Control". In: *IEEE Trans. Pattern Analysis and Machine Intelligence*.

Deisenroth, Marc and Carl E. Rasmussen (2011). "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: *Proceedings of the 28th International Conference on Machine Learning*.

Depeweg, Stefan, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft (2018). "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning". In: *Proceedings of the 35th International Conference on Machine Learning*.

Doerr, Andreas, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian (2018). "Probabilistic Recurrent State-Space Models". In: *Proceedings of the 35th International Conference on Machine Learning*.

Durstewitz, Daniel (2017). "A State Space Approach for Piecewise-linear Recurrent Neural Networks for Identifying Computational Dynamics from Neural Measurements". In: *PLoS Computational Biology*.

Dziugaite, Gintare and Daniel M. Roy (2017). "Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data". In: *Uncertainty in Artificial Intelligence*.

Ebert, Sandra, Mario Fritz, and Bernt Schiele (2012). "RALF: A Reinforced Active Learning Formulation for Object Class Recognition". In: *Conference on Computer Vision and Pattern Recognition*.

Efron, Bradley (2012). *Large-scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. Cambridge University Press.

Fang, Meng, Yuan Li, and Trevor Cohn (2017). "Learning how to Active Learn: A Deep Reinforcement Learning Approach". In: *Empirical Methods in Natural Language Processing*.

Figurnov, Mikhail, Shakir Mohamed, and Andriy Mnih (2018). "Implicit Reparameterization Gradients". In: *Advances in Neural Information Processing Systems*.

Fortuin, Vincent, Adrià Garriga-Alonso, Florian Wenzel, Gunnar Ratsch, Richard E Turner, Mark van der Wilk, and Laurence Aitchison (2020). "Bayesian Neural Network Priors Revisited". In: *"I Can't Believe It's Not Better!"NeurIPS 2020 workshop*.

Frey, Brendan J. and Geoffrey E. Hinton (1999). "Variational Learning in Nonlinear Gaussian Belief Networks". In: *Neural Computation*.

G. Matthews, Alexander G. de, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani (2018). "Gaussian Process Behaviour in Wide Deep Neural Networks". In: *International Conference on Learning Representations*.

Gal, Yarin and Zoubin Ghahramani (2016a). "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference". In: *4th International Conference on Learning Representations (ICLR) workshop track*.

– (2016b). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of The 33rd International Conference on Machine Learning*.

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani (2017). "Deep Bayesian Active Learning with Image Data". In: *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*.

Gan, Zhe, Chunyuan Li, Ricardo Henao, David E. Carlson, and Lawrence Carin (2015). "Deep Temporal Sigmoid Belief Networks for Sequence Modeling". In: *Advances in Neural Information Processing Systems*.

Garnelo, Marta, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami (2018a). "Conditional Neural Processes". In: *Proceedings of the 35th International Conference on Machine Learning*.

Garnelo, Marta, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, SM Ali Eslami, and Yee Whye Teh (2018b). "Neural Processes". In: *arXiv preprint arXiv:1807.01622*.

Garriga-Alonso, Adrià, Carl Edward Rasmussen, and Laurence Aitchison (2019). "Deep Convolutional Networks as shallow Gaussian Processes". In: *International Conference on Learning Representations*.

Gast, Jochen and Stefan Roth (2018). "Lightweight Probabilistic Deep Networks". In: *Conference on Computer Vision and Pattern Recognition*.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin (2013). *Bayesian Data Analysis*. CRC press.

Germain, Pascal, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien (2016). "PAC-Bayesian Theory Meets Bayesian Inference". In: *Advances in Neural Information Processing Systems*.

Germain, Pascal, Alexandre Lacasse, François Laviolette, and Mario Marchand (2009). "PAC-Bayesian Learning of Linear Classifiers". In: *International Conference on Machine Learning*.

Ghahramani, Zoubin (2015). "Probabilistic Machine Learning and Artificial Intelligence". In: *Nature*.

Ghosh, Soumya and Finale Doshi-Velez (2017). "Model Selection in Bayesian Neural Networks via Horseshoe Priors". In: *arXiv preprint arXiv:1705.10388*.

Ghosh, Soumya, Francesco Maria Delle Fave, and Jonathan Yedidia (2016). "Assumed Density Filtering Methods for Scalable Learning of Bayesian Neural Networks". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press.

Goodfellow, Ian, Jonathan Shlens, and Christian Szegedy (2015). "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations*.

Graves, Alex (2011). "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems*.

Guedj, Benjamin (2019). "A primer on PAC-Bayesian learning". In: *arXiv preprint arXiv:1901.05353*.

Haußmann, Manuel, Sebastian Gerwinn, and Melih Kandemir (2020a). "Bayesian Evidential Deep Learning with PAC Regularization". In: *Third Symposium on Advances in Approximate Bayesian Inference*.

Haußmann, Manuel, Sebastian Gerwinn, Andreas Look, Barbara Rakitsch, and Melih Kandemir (2021). "Learning Partially Known Stochastic Dynamics with Empirical PAC Bayes". In: *Proceedings of the Twenty-Fourth International Conference on Artificial Intelligence and Statistics*.

Haußmann, Manuel, Fred A. Hamprecht, and Melih Kandemir (2019). "Deep Active Learning with Adaptive Acquisition". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*.

– (2020b). "Sampling-Free Variational Inference of Bayesian Neural Networks by Variance Backpropagation". In: *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proceedings of the IEEE International Conference on Computer Vision*.

– (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hegde, Pashupati, Markus Heinonen, Harri Lähdesmäki, and Samuel Kaski (2019). "Deep learning with differential Gaussian process flows". In: *Proceedings of Machine Learning Research*.

Heinonen, Markus, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki (2018). "Learning unknown ODE models with Gaussian processes". In: *Proceedings of the 35th International Conference on Machine Learning*.

Hensman, James, Nicolò. Fusi, and Neil D. Lawrence (2013). "Gaussian Processes for Big Data". In: *Uncertainty in Artificial Intelligence*.

Hernández-Lobato, Jose Miguel and Ryan Adams (2015). "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks". In: *Proceedings of the 32nd International Conference on Machine Learning*.

Hinton, Geoffrey E. and Drew van Camp (1993). "Keeping Neural Networks Simple". In: *International Conference on Artificial Neural Networks*.

Houlsby, Neil, Ferenc Huszar, Zoubin Ghahramani, and Jose Hernández-lobato (2012). "Collaborative Gaussian Processes for Preference Learning". In: *Advances in Neural Information Processing Systems*.

Hron, Jiri, Alexander G. de G. Matthews, and Zoubin Ghahramani (2017). "Variational Gaussian Dropout is not Bayesian". In: *Bayesian Deep Learning workshop*.

Hsu, Wei-Ning and Hsuan-Tien Lin (2015). "Active Learning by Learning." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Huang, Gao, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger (2017). "Densely Connected Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*.

Isensee, Fabian, Paul F. Jaeger, Simon A.A. Kohl, Jens Petersen, and Klaus H. Maier-Hein (2020). "nnU-Net: a Self-configuring Method for Deep Learning-based Biomedical Image Segmentation". In: *Nature Methods*.

Jang, Eric, Shixiang Gu, and Ben Poole (2016). "Categorical Reparameterization with Gumbel-Softmax". In: *International Conference on Learning Representations*.

Jumper, John, R. Evans, A. Pritzel, T. Green, M. Figurnov, K. Tunyasuvunakool, O. Ronneberger, R. Bates, A. Zidek, A. Bridgland, et al. (2020). "High Accuracy Protein Structure Prediction using Deep Learning". In: *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*.

Kandemir, Melih (2018). "Variational Closed-form Deep Neural Net Inference". In: *Pattern Recognition Letters*.

Kantas, Nikolas, Arnaud Doucet, S. Sing Singh, Jan Maciejowski, and Nicola Chopin (2015). "On particle methods for parameter estimation in state-space models". In: *Statistical Science*.

Kass, Robert E. and Adrian E. Raftery (1995). "Bayes Factors". In: *Journal of the American Statistical Association*.

Kendall, Alex and Yarin Gal (2017). "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems*.

Kingma, Diederik P and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*.

Kingma, Diederik P. and Max Welling (2014). "Auto-encoding Variational Bayes". In: *Proceedings of the 2nd International Conference on Learning Representations*.

Kingma, Durk P. and Prafulla Dhariwal (2018). "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*.

Kingma, Durk P., Tim Salimans, and Max Welling (2015). "Variational Dropout and the Local Reparameterization Trick". In: *Advances in Neural Information Processing Systems*.

Klambauer, Günter, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter (2017). "Self-normalizing Neural Networks". In: *Advances in Neural Information Processing Systems*.

Kloeden, Peter E. and Eckhard Platen (2011). *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag.

Knoblauch, Jeremias, Jack Jewson, and Theodoros Damoulas (2019). "Generalized Variational Inference: Three Arguments for Deriving new Posteriors". In: *arXiv preprint arXiv:1904.02063*.

Konyushkova, Ksenia, Raphael Sznitman, and Pascal Fua (2017). "Learning Active Learning from Data". In: *Advances in Neural Information Processing Systems*.

Krizhevsky, Alex and Geoffrey Hinton (2009). *Learning multiple layers of features from tiny images*. Tech. rep.

Lampinen, Jouko and Aki Vehtari (2001). "Bayesian approach for neural networks—review and case studies". In: *Neural Networks*.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE*.

Lee, Jaehoon, Jascha Sohl-Dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri (2018a). "Deep Neural Networks as Gaussian Processes". In: *International Conference on Learning Representations*.

Lee, Juho, Saehoon Kim, Jaehong Yoon, Hae Beom Lee, Eunho Yang, and Sung Ju Hwang (2018b). "Adaptive Network Sparsification with Dependent Variational Beta-Bernoulli Dropout". In: *arXiv preprint arXiv:1805.10896*.

Li, Xuechen, Yi Wu, Lester Mackey, and Murat A Erdogdu (2019). "Stochastic Runge-Kutta Accelerates Langevin Monte Carlo and Beyond". In: *Advances in Neural Information Processing Systems*.

Li, Yingzhen and Yarin Gal (2017). "Dropout Inference in Bayesian Neural Networks with Alpha-divergences". In: *Proceedings of the 34th International Conference on Machine Learning*.

Li, Yingzhen and Richard E Turner (2016). "Rényi Divergence Variational Inference". In: *Advances in Neural Information Processing Systems*.

Look, Andreas and Melih Kandemir (2019). "Differential Bayesian Neural Nets". In: *4th NeurIPS Workshop on Bayesian Deep Learning*.

Louizos, Christos, Karen Ullrich, and Max Welling (2017). "Bayesian Compression for Deep Learning". In: *Advances in Neural Information Processing Systems*.

Louizos, Christos and Max Welling (2016). "Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors". In: *Proceedings of The 33rd International Conference on Machine Learning*.

– (2017). "Multiplicative Normalizing Flows for Variational Bayesian Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*.

Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng (2013). "Rectifier nonlinearities improve neural network acoustic models". In: *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*.

MacKay, David JC (1992). "A Practical Bayesian Framework for Backpropagation Networks". In: *Neural Computation*.

– (1995). "Probable Networks and Plausible Predictions – A Review of Practical Bayesian Methods for Supervised Neural Networks". In: *Network: Computation in Neural Systems*.

– (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge university press.

Maddison, Chris J., Andriy Mnih, and Yee Whye Teh (2016). "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: *International Conference on Learning Representations*.

Malinin, Andrey and Mark Gales (2018). "Predictive Uncertainty Estimation via Prior Networks". In: *Advances in Neural Information Processing Systems*.

Maurer, Andreas (2004). "A Note on the PAC Bayesian Theorem". In: *arXiv preprint cs/0411099*.

McAllester, David (1998). "Some PAC-Bayesian Theorems". In: *Proceedings of the Eleventh Annual Conference In Computational Learning Theory*.

– (1999). "PAC-Bayesian Model Averaging". In: *Conference In Computational Learning Theory*.

– (2003). "PAC-Bayesian Stochastic Model Selection". In: *Machine Learning*.

McElreath, Richard (2020). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. CRC press.

Miller, Andrew, Nick Foti, Alexander D'Amour, and Ryan P Adams (2017). "Reducing Reparameterization Gradient Variance". In: *Advances in Neural Information Processing Systems*.

Minka, Thomas P. (2001). "A Family of Algorithms for Approximate Bayesian Inference". PhD thesis. Massachusetts Institute of Technology.

– (2013). "Expectation Propagation for Approximate Bayesian Inference". In: *Conference on Uncertainty in Artificial Intelligence*.

Molchanov, Dmitry, Arsenii Ashukha, and Dmitry Vetrov (2017). "Variational Dropout Sparsifies Deep Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*.

Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT press.

Neal, Radford (1995). "Bayesian Learning for Neural Networks". PhD thesis.

Neal, Radford (2010). "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo*.

Neklyudov, Kirill, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P. Vetrov (2017). "Structured Bayesian Pruning via Log-Normal Multiplicative Noise". In: *Advances in Neural Information Processing Systems*.

Øksendal, Bernt (1992). *Stochastic Differential Equations: An Introduction with Applications*. Springer-Verlag.

Pang, Kunkun, Mingzhi Dong, Yang Wu, and Timothy Hospedales (2018). "Meta-learning transferable active learning policies by deep reinforcement learning". In: *arXiv preprint arXiv:1806.04798*.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*.

Qiu, Zhicong, David J. Miller, and George Kesidis (2017). "A Maximum Entropy Framework for Semisupervised and Active Learning With Unknown and Label-Scarce Classes". In: *IEEE Transactions on Neural Networks and Learning Systems*.

Quionero-Candela, Joaquin, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence (2009). *Dataset Shift in Machine Learning*. The MIT Press.

Rackauckas, Christopher, Yingbo Ma, Julius Martensen, Collin Warner, Krill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman (2020). "Universal differential equations for scientific machine learning". In: *arXiv preprint arXiv:2001.04385*.

Rasmussen, Carl Edward and Zoubin Ghahramani (2001). "Occam's Razor". In: *Advances in Neural Information Processing Systems*.

Rasmussen, Carl Edward and Christopher KI Williams (2006). *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA.

Redmon, Joseph and Ali Farhadi (2018). "YOLOv3: An Incremental Improvement". In: *arXiv preprint arXiv:1804.02767*.

Reeb, David, Andreas Doerr, Sebastian Gerwinn, and Barbara Rakitsch (2018). "Learning Gaussian Processes by Minimizing PAC-Bayesian Generalization Bounds". In: *Advances in Neural Information Processing Systems*.

Rezende, Danilo and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning*.

Rezende, Danilo, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31st International Conference on Machine Learning*.

Robert, Christian and George Casella (2013). *Monte Carlo Statistical Methods*. Springer Science & Business Media.

Roeder, Geoffrey, Yuhuai Wu, and David K. Duvenaud (2017). "Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference". In: *Advances in Neural Information Processing Systems*.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International*

*Conference on Medical image computing and computer-assisted intervention*. Springer.

Särkkä, Simo. and Arno Solin (2019). *Applied Stochastic Differential Equations*. Cambridge University Press.

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz (2015). "Trust Region Policy Optimization". In: *Proceedings of the 32nd International Conference on Machine Learning*.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). "Proximal Policy Optimization Algorithms". In: *arXiv preprint arXiv:1707.06347*.

Seeger, Matthias (2002). "PAC-Bayesian Generalisation Error Bounds for Gaussian Process Classification". In: *Journal of Machine Learning Research*.

Sener, Ozan and Silvio Savarese (2018). "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: *International Conference on Learning Representations*.

Sensoy, Murat, Lance Kaplan, and Melih Kandemir (2018). "Evidential Deep Learning to Quantify Classification Uncertainty". In: *Advances in Neural Information Processing Systems*.

Settles, Burr (2012). *Active Learning*. Morgan & Claypool Publishers.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. (2018). "A General Reinforcement Learning Algorithm that masters Chess, Shogi, and Go through Self-play". In: *Science*.

Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams (2012). "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*.

Springenberg, Jost Tobias, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller (2015). "Striving for Simplicity: The All Convolutional Net". In: *Workshop on the International Conference on Learning Representations*.

Springenberg, Jost Tobias, Aaron Klein, Stefan Falkner, and Frank Hutter (2016). "Bayesian Optimization with Robust Bayesian Neural Networks". In: *Advances in Neural Information Processing Systems*.

Srinivas, Niranjan, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger (2012). "Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting". In: *IEEE Transactions on Information Theory*.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2016). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research*.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement Learning: An Introduction*. MIT press.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus (2014). "Intriguing Properties of Neural Networks". In: *International Conference on Learning Representations*.

The Theano Development Team (2016). "Theano: A Python framework for fast computation of mathematical expressions". In: *arXiv preprint arXiv:1605.02688*.

Tolstikhin, Ilya O. and Yevgeny Seldin (2013). "PAC-Bayes-Empirical-Bernstein Inequality". In: *Advances in Neural Information Processing Systems*.

Tucker, George, Dieterich Lawson, Shixiang Gu, and Chris J. Maddison (2018). "Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives". In: *International Conference on Learning Representations*.

Tucker, George, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein (2017). "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models". In: *Advances in Neural Information Processing Systems*.

Tzen, Belinda and Maxim Raginsky (2019). "Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit". In: *arXiv preprint arXiv:1905.09883*.

Valiant, Leslie G (1984). "A Theory of the Learnable". In: *Communications of the ACM*.

Wainwright, Martin J. and Michael I. Jordan (2008). "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends in Machine Learning*.

Wang, Keze, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin (2016). "Cost-effective Active Learning for Deep Image Classification". In: *IEEE Transactions on Circuits and Systems for Video Technology*.

Wang, Sida and Christopher Manning (2013). "Fast Dropout Training". In: *Proceedings of the 30th International Conference on Machine Learning*.

Wasserman, Larry (2013). *All of Statistics: A Concise Course in Statistical Inference*. Springer Science & Business Media.

Welling, Max and Yee Whyee Teh (2011). "Bayesian Learning via Stochastic Gradient Langevin Dynamics". In: *Proceedings of the 28th International Conference on Machine Learning*.

Williams, Ronald J. (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine learning*.

Wu, Anqi, Sebastian Nowozin, Edward Meeds, Richard E. Turner, Jose Miguel Hernandez-Lobato, and Alexander L. Gaunt (2019). "Deterministic Variational Inference for Robust Bayesian Neural Networks". In: *International Conference on Learning Representations*.

Wu, Yuxin and Kaiming He (2018). "Group Normalization". In: *Proceedings of the European Conference on Computer Vision*.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *arXiv preprint arXiv:1708.07747*.

Yildiz, Cagatay, Markus Heinonen, and Harri Lahdesmaki (2019). "ODE2VAE: Deep generative second order ODEs with Bayesian neural networks". In: *Advances in Neural Information Processing Systems*.

Yu, Fisher, Vladlen Koltun, and Thomas Funkhouser (2017). "Dilated Residual Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

## COLOPHON

This document was typeset using the typographical look-and-feel `classic-thesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*.