Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Jakub Świątkowski
Nr alb: 416261

Variational Inference Applications in Deep Learning

Rozprawa doktorska
w dziedzinie nauk ścisłych i przyrodniczych
w dyscyplinie informatyka

Praca wykonana pod kierunkiem
dr hab. Marka Cygana, prof. UW
Instytut Informatyki

Warszawa, Wrzesień, 2023

Oświadczenie kierującego pracą

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie stopnia doktora w dziedzinie nauk ścisłych i przyrodniczych w dyscyplinie informatyka.

Data Podpis kierującego pracą

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza rozprawa doktorska została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiazującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem stopnia doktora w innej jednostce.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Słowa kluczowe

Inferencja wariacyjna, metody aproksymacji, bayesowskie sieci neuronowe, zmienne ukryte, uczenie reprezentacji, głębokie sieci neuronowe, zanurzenia w głębokich sieciach neurownowych, modele generatywne, synteza mowy, dubbing maszynowy, transfer prozodii mowy, wielo-językowość syntezy mowy

Tytuł pracy w języku polskim

Zastosowania inferencji wariacyjnej w głębokich sieciach neuronowych

Streszczenie w języku polskim

Ta praca doktorska bada przecięcie uczenia głębokiego i wnioskowania wariacyjnego, dwóch znaczących obszarów w dziedzinie uczenia maszynowego i statystyki. Choć modele uczenia głębokiego wykazały wyjątkową wydajność w różnych aplikacjach, ich ograniczenia w kwantyfikacji niepewności i probabilistycznych prognozach stanowią znaczne wyzwania. Aby je pokonać, niniejsze badania wykorzystują zasady wnioskowania bayesowskiego, aby wyposażyć modele uczenia głębokiego w niezawodność, interpretowalność i rozumowanie probabilistyczne. Jednak bezpośrednie zastosowanie wnioskowania bayesowskiego w skomplikowanych modelach, takich jak sieci uczenia głębokiego, jest niemożliwe do przeprowadzenia z powodu problemów obliczeniowych i skalowalności. Aby to rozwiązać, praca korzysta z wnioskowania wariacyjnego jako techniki aproksymacyjnej. Konkretnie, w pracy badamy zastosowania wnioskowania wariacyjnego do bayesowskich sieci neuronowych i syntezy mowy przy użyciu sieci neuronowych.

W kontekście bayesowskich sieci neuronowych (BNNs), wnioskowanie wariacyjne jest stosowane do wag sieci. Badania prezentowane w tej pracy znacznie rozszerzają nasze zrozumienie i wykorzystanie BNNs. Odkryto, że powszechnie używana forma wnioskowania wariacyjnego w BNNs wykazuje strukturę niskiego rzędu. Odkrycie to pozwala na redukcję liczby parametrów i prowadzi do przyspieszenia uczenia modelu. Ponadto, badanie prezentuje intrygujące stwierdzenie, że BNNs odbiegające od teoretycznie optymalnego wnioskowania Bayesa wykazują lepszą wydajność. Szereg hipotez jest systematycznie badanych, aby wyjaśnić to zaobserwowane zachowanie.

W dziedzinie syntezy mowy, wnioskowanie wariacyjne jest stosowane do nauki rozłącznych, interpretowalnych i kontrolowalnych reprezentacji danych za pomocą Auto-Enkoderów Wariacyjnych (VAEs). Praca proponuje nową metodę dubbingu maszynowego, która uczy się zanurzeń prozodii mowy, które są rozłączone od języka, mówcy i szumów kanału, dzięki czemu mogą być przeniesione między językami i mówcami, aby generować czyste dubbingi maszynowe. Ponadto, badanie pokazuje, że modelowanie i transfer prozodii na poziomie frazy prozodycznej, zamiast na poziomie całej wypowiedzi, znacznie poprawia jakość dubbingu maszynowego.

Podsumowując, poprzez połączenie wnioskowania wariacyjnego z uczeniem głębokim, ta praca nie tylko zwiększa wydajność i interpretowalność modeli, ale także przesuwa granice możliwego w dziedzinach takich jak bayesowskie sieci neuronowe i neuralna synteza mowy. Badanie dostarcza nowych spostrzeżeń i metod, które torują drogę dla przyszłych rozwojów w tych obszarach.

Streszczenie w języku angielskim

This PhD thesis explores the intersection of deep learning and variational inference, two significant areas in machine learning and statistics. While deep learning models have demonstrated exceptional performance across a range of applications, their inherent limitations in uncertainty quantification and probabilistic predictions present significant challenges. To overcome these challenges, this research leverages the principles of Bayesian inference to imbue deep learning models with robustness, interpretability, and probabilistic reasoning. However, direct application of Bayesian inference in complex models such as deep learning networks is intractable due to computational and scalability issues. To address this, the thesis employs variational inference as an approximation technique. This work specifically explores the application of variational inference to Bayesian neural networks and neural speech synthesis.

In the context of Bayesian neural networks (BNNs), variational inference is applied to the weights of the network. The research presented in this thesis significantly enhances our understanding and utilization of BNNs. It uncovers that a commonly used form of variational inference in BNNs exhibits a low-rank structure. This discovery allows for a reduction in the number of parameters and leads to accelerated model training. Additionally, the research presents an intriguing finding that BNNs deviating from the theoretically optimal Bayes inference show improved performance. A series of hypotheses are systematically examined to explain this observed behaviour.

In the domain of speech synthesis, variational inference is applied to learning disentangled, interpretable, and controllable latent representations of data using Variational Auto-Encoders (VAEs). The thesis proposes a novel method for machine dubbing that learns speech prosody embeddings, which are disentangled from language, speaker, and channel noise and can be transferred across languages and speakers to generate clean machine dubs. Moreover, the research demonstrates that modelling and transferring prosody at a prosodic phrase level, instead of a whole utterance level, significantly improves machine dubbing quality.

In summary, by intertwining variational inference with deep learning, this thesis not only enhances model performance and interpretability but also pushes the boundaries of fields such as Bayesian neural networks and neural speech synthesis. This research provides novel insights and methods that pave the way for future developments in these areas.

Contents

Chapter 1

Introduction

This PhD thesis explores applications of variational inference (Peterson, 1987; Hinton & Van Camp, 1993a) techniques in the domain of deep learning (Goodfellow et al., 2016). Variational inference provides a powerful framework for approximate Bayesian inference (Pearl, 1988; Murphy, 2012), allowing for efficient and scalable probabilistic modelling. In recent years, deep learning has emerged as a dominant approach for solving complex tasks in various domains. However, incorporating uncertainty estimation and probabilistic modelling into deep learning systems remains a challenging problem. This thesis investigates the potential of variational inference in addressing these challenges and explores its applications in different areas of deep learning. The research contributes novel methodologies, theoretical insights, and empirical evaluations, shedding light on the benefits, limitations, and future directions of variational inference in deep learning.

1.1 List of publications

My thesis comprises four published papers from P1 to P4:

P1 **J. Swiatkowski**, K. Roth, B. Veeling, L. Tran, J. Dillon, J. Snoek, S. Mandt, T. Salimans, R. Jenatton, S. Nowozin
"The k-tied Normal Distribution: A Compact Parameterization of Gaussian Mean Field Posteriors in Bayesian Neural Networks"
International Conference on Machine Learning 2020
CORE Rank: A*, MNiSW: 200
As the sole first author of this work my contributions cover all the major work items including proposing and developing a research idea, method implementation, experimentation, analysis, and paper writing. Other co-authors contributed through discussions, final paper writing (abstract, introduction and related work), and the proof of the Matrix Variate Normal Parameterization.
I estimate my contribution to be at 80%.

P2 F. Wenzel, K. Roth, B. Veeling, **J, Swiatkowski**, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, S. Nowozin
"How Good is the Bayes Posterior in Deep Neural Networks Really?"
International Conference on Machine Learning 2020
CORE Rank: A*, MNiSW: 200
As a secondary author of this work, my contribution is mainly through discussions and my initial observation of the cold posterior effect in the variational inference variant of Bayesian neural networks, which at least partially motivated the research done in this work. Nevertheless, I include this work due to its significant impact on the field of machine learning, especially related to Bayesian neural networks (284 citations until August 2023).

P3 **J. Swiatkowski**, D. Wang, M. Babianski, P. Tobing, R. Vipperla, V. Pollet
"Cross-lingual Prosody Transfer for Expressive Machine Dubbing"
INTERSPEECH 2023
CORE Rank: A, MNiSW: 140
As one of three first authors of this work, my contributions cover proposing and developing the research idea (proposal of the cross-lingual prosody transfer through reference encoder), initial method implementation and experimentation showing the success of such transfer, method evaluation, paper writing.
I estimate my contribution to be at 35%.

P4 **J. Swiatkowski**, D. Wang, M. Babianski, G. Coccia, P. Tobing, R. Vipperla, V. Klimkov, V. Pollet
"Expressive Machine Dubbing Through Phrase-level Cross-lingual Prosody Transfer"
INTERSPEECH 2023
CORE Rank: A, MNiSW: 140
My contributions cover proposing and developing the research idea (proposal of the phrase-level cross-lingual prosody transfer), method implementation, experimentation, analysis, and paper writing.
I estimate my contribution to be at 60%.

1.2 Overview

Deep learning has revolutionized the field of artificial intelligence, enabling significant advancements in various domains, including computer vision, natural language processing, and speech processing. Deep neural networks have demonstrated remarkable performance in complex tasks, surpassing traditional machine learning methods in many areas. However, deep learning models often lack the ability to capture uncertainty and provide probabilistic predictions, which is essential for decision-making under uncertainty, robustness analysis, and reliable risk assessment.

In contrast, Bayesian inference provides a principled framework for probabilistic modelling and uncertainty quantification. It allows for incorporating prior knowledge, learning from limited data, and obtaining posterior distributions over model parameters. Bayesian methods have been widely studied and applied in various fields, but their direct integration with deep learning models is challenging due to computational complexity and scalability issues.

Variational inference, a branch of Bayesian inference, offers a promising approach to address these challenges. It provides a scalable and efficient approximation technique for Bayesian inference, enabling the incorporation of uncertainty estimation and probabilistic modelling in deep learning models. It estimates and manipulates uncertainty by treating specific model components as latent variables. In the context of deep learning, the latents can be neural network weights or learned data representations.

Traditionally, neural network weights are treated as fixed parameters and learned using optimization techniques like stochastic gradient descent. However, this approach does not account for uncertainty in the weights, which can be valuable in various scenarios, such as model generalization, robustness, and interpretability.

By treating neural network weights as variational inference latents, we can capture and quantify uncertainty in the weights, leading to more robust and interpretable models. Bayesian neural networks (BNNs) (Peterson, 1987; Hinton & Van Camp, 1993a; Blei et al., 2017) are a prime example of this approach, where the posterior distribution over the weights is approximated using variational inference. BNNs provide a means to encode uncertainty in the model's predictions, enabling robust decision-making and confidence estimation. This describes two published papers advancing the state-of-the-art and understanding of Bayesian neural networks co-authored by Jakub Swiatkowski, the author of this thesis.

In the first paper (P1), where Jakub Swiatkowski is the main author, two main contributions are:

1. Finding that the most commonly used family of variational posteriors for Bayesian neural network weights has a low-rank structure.

2. Proposed low-rank parametrization of the variational posteriors, which reduces the number of parameters by half and leads to faster training convergence.

In the second paper (P2), where Jakub Swiatkowski is a secondary author, our contributions are:

1. We demonstrate that artificially reducing the variance of learned BNN posteriors improves predictive performance over theoretically optimal Bayes predictive posteriors in BNNs.

2. We put forth and systematically examine hypotheses that could explain the observed behaviour.

3. We introduce two new diagnostic tools for assessing the approximation quality of the learned posteriors.

In addition to modelling uncertainty in neural network weights, variational inference can also be applied to learning data representations using Variational Auto-Encoders (VAEs) (Kingma & Welling, 2013; Salimans et al., 2013; Rezende & Mohamed, 2015). Learned representations capture the underlying structure and features of the data, creating more expressive and compact features for downstream tasks. By treating the learned data representations as variational inference latents, we can learn and sample from rich distributions over these representations, providing a more flexible, controllable and interpretable model.

One practical application that benefits from such learned representations using VAEs is speech synthesis (Zhang et al., 2019c; Kim et al., 2021). Speech synthesis aims to generate natural and human-like speech from textual or other input representations. By leveraging variational inference to learn latent representations, we can capture the complex dependencies and variability in speech data, improving synthesis quality, controllability and expressiveness.

In speech synthesis, the ability to model uncertainty in latent representations allows for more robust and contextually relevant speech generation. By manipulating the latent variables, we can control specific speech attributes, such as speaking style, emotion, or intonation, enabling personalized and expressive speech synthesis (Zhang et al., 2019c). Moreover, variational inference facilitates the learning of disentangled representations, separating different factors of variation, such as speaker identity and linguistic content, into distinct dimensions of the latent space (Kumar et al., 2018; Locatello et al., 2019).

Controllability in speech synthesis is especially crucial for tasks like machine dubbing. In machine dubbing, the goal is to synthesize speech for translated text while maintaining the vocal performance of the original language's voice recording. We describe two published papers showing an application of VAEs in this setting, where Jakub Swiatkowski is the first author.

In the first dubbing paper (P3), our contributions are:

1. We present a novel method capable of learning VAE representations of vocal performance with disentangled language and voice characteristics that can be transferred across languages.

2. We propose combining multiple VAE encoders that disentangle vocal performance and background noise allowing for clean speech synthesis from a noisy reference audio.

In the second dubbing paper (P4), our contributions are:

1. We extend the previous paper with a more granular, phrase-level, VAE-learned representation of the vocal performance. We show that such representation results in better dubbing quality than the utterance-level representations proposed in P3.

2. We propose a length-based regularization for the phrase-level representation that improves the disentanglement of vocal performance from the original language content.

By intertwining variational inference with deep learning, this thesis not only enhances model performance and interpretability but also pushes the boundaries of fields such as

Bayesian neural networks and neural speech synthesis. In the remainder of this chapter, we will introduce the concepts of deep learning, Bayesian inference and variational inference. In Chapters 2 and 3, we will delve into the details of applying variational inference in Bayesian neural networks and neural speech synthesis respectively. In both chapters, we provide summaries of the relevant publications (P1-P4), which are the main body of this thesis in Chapters 4–7.

1.3 Deep learning

Deep Learning (Goodfellow et al., 2016), a subfield of machine learning, has achieved remarkable success across a wide array of tasks, including image classification (Krizhevsky et al., 2012), speech recognition (Graves et al., 2013), natural language processing (Vaswani et al., 2017), and even board and video games (Silver et al., 2016). Deep learning algorithms, commonly referred to as neural networks, are models comprised of many processing layers (hence "deep"), which learn representations of data with multiple levels of abstraction.

Neural networks, in particular, are comprised of interconnected nodes, known as neurons, organized into layers. The input is passed through these layers, transformed at each stage, and ultimately generates an output. The primary strength of deep learning lies in the fact that these networks can learn directly from raw data and automatically extract useful features, mitigating the need for handcrafted feature extraction, which is often labor-intensive and task-specific.

The success of deep learning models is attributed to their capacity to learn complex patterns and interactions within data. These models are trained through a process known as backpropagation (Rumelhart et al., 1986), an algorithm used for adjusting the model's weights to minimize the difference between the model's predictions and the actual data. While the initial layers of a deep learning model might learn simple patterns, the more advanced layers combine these simpler patterns to understand more complicated relationships.

Deep learning models' flexibility and strength are especially apparent in tasks involving unstructured data (e.g., images, audio, and text), where traditional machine learning models struggle. Deep learning can handle these types of data due to its ability to learn hierarchical features and deal with high-dimensional data. However, while deep learning has had significant success, there are still numerous challenges, such as overfitting, interpretability, and the need for large amounts of training data. This thesis aims to contribute towards addressing some of these challenges in the context of Bayesian deep learning and speech synthesis.

In the following sections, we will delve into the Bayesian inference method, one of the main concepts used in this work, and how it can be combined with deep learning to better model uncertainty, regularize models, and improve their interpretability.

1.4 Bayesian inference

Bayesian inference (Pearl, 1988; Murphy, 2012) is a powerful framework for reasoning and making predictions under uncertainty. It provides a systematic way to update our beliefs or

knowledge about a particular phenomenon as new evidence becomes available (Gelman et al., 2013). Named after the Reverend Thomas Bayes, an 18th-century English statistician and philosopher, Bayesian inference is a cornerstone of modern statistics and has applications in a wide range of fields, including machine learning, artificial intelligence, economics, and healthcare (McGrayne, 2011).

At its core, Bayesian inference is based on Bayes' theorem (2.1), which describes how to update the probability of a hypothesis (or a set of hypotheses) $\mathbf{z}$ given new evidence $\mathbf{x}$. The theorem mathematically connects the prior probability $p(\mathbf{z})$, which represents our initial beliefs or knowledge about the hypothesis $\mathbf{z}$, with the likelihood $p(\mathbf{x}|\mathbf{z})$, which quantifies how well the observed evidence supports or contradicts the hypothesis. By combining the prior probability with the likelihood, Bayes' theorem allows us to compute the posterior probability $p(\mathbf{z}|\mathbf{x})$, which represents our updated belief about the hypothesis in light of the new evidence.

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \tag{1.1}$$

The key idea in Bayesian inference is to treat probabilities as measures of our own uncertainty or degrees of belief. Unlike frequentist statistics, which often focus on the long-run properties of estimators, Bayesian inference directly addresses our subjective beliefs and updates them in a rational and coherent manner (Jaynes, 2003). This subjective interpretation of probability allows us to incorporate prior knowledge, expert opinions, or subjective judgments into the inference process (Berger, 2013).

One of the strengths of Bayesian inference is its ability to handle complex problems involving multiple parameters or hypotheses. By using probability distributions to represent uncertainty, we can express our beliefs about the values of these parameters or hypotheses and update them as new data becomes available. This flexibility enables Bayesian inference to handle a wide range of models, from simple linear regression to complex hierarchical models with latent variables (Gelman et al., 2013).

Posterior distribution allows us to integrate out the latent variables of our model $\mathbf{z}$ and estimate the probability of new data $\mathbf{x}^*$ after observing all the previous evidence $\mathbf{x}$:

$$p(\mathbf{x}^*|\mathbf{x}) = \int p(\mathbf{x}^*|\mathbf{z})p(\mathbf{z}|\mathbf{x})\, d\mathbf{z} \tag{1.2}$$

In addition to its theoretical elegance, Bayesian inference offers practical advantages. It provides a unified framework for data analysis, allowing us to seamlessly combine prior knowledge with observed data (Box & Tiao, 2011). Furthermore, Bayesian methods naturally quantify uncertainty by providing posterior probability distributions, which can be used to estimate credible intervals or perform decision analysis (Robert et al., 2007).

While Bayesian inference offers numerous benefits, it also presents challenges. Computing the posterior distribution can be analytically intractable for complex models. In particular, computing the normalizing constant of the exact posterior $p(\mathbf{x})$ involves integration over all possible latent variables that is computationally prohibitive for the complex models.

This motivates the use of approximation techniques such as Markov Chain Monte Carlo (MCMC) (Gelfand & Smith, 1990) or Variational Inference (Peterson, 1987; Hinton & Van Camp, 1993a; Jordan et al., 1999). We focus on the latter in this thesis. Additionally, the choice of prior distributions can have a significant impact on the results, and different priors may lead to different conclusions. Therefore, careful consideration and sensitivity analysis are necessary when specifying prior beliefs, as demonstrated in Chapter 5 of this thesis.

1.5 Variational inference

Variational Inference (Peterson, 1987; Hinton & Van Camp, 1993a; Jordan et al., 1999) is a family of methods used to approximate complex posterior distributions when direct computation is intractable. It offers a scalable and computationally efficient alternative to exact Bayesian inference, allowing us to perform inference in large-scale models.

The main idea behind variational inference is to cast the problem of approximating the posterior distribution as an optimization task. Instead of directly calculating the true posterior, variational inference introduces a simpler family of distributions, called the variational family, from which we seek the best approximation.

The variational family is typically defined by a set of parameters that can be optimized to minimize the divergence between the true posterior and the approximating distribution. The optimization problem is often formulated as minimizing the Kullback-Leibler (KL) divergence between the true posterior and the variational distribution. In particular, we minimize the KL divergence D_{KL} between the variational distribution $q_{\boldsymbol{\theta}}(\mathbf{z})$ and the true posterior distribution $p(\mathbf{z}|\mathbf{x})$, which is given by

$$\begin{aligned} D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{z})||p(\mathbf{z}|\mathbf{x})] &= \mathbb{E}_q\left[\log \frac{q_{\boldsymbol{\theta}}(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}\right] \\ &= \mathbb{E}_q\left[\log \frac{q_{\boldsymbol{\theta}}(\mathbf{z})}{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})/p(\mathbf{x})}\right] \\ = \mathbb{E}_q\left[\log \frac{q_{\boldsymbol{\theta}}(\mathbf{z})}{p(\mathbf{z})}\right] - \mathbb{E}_q\left[\log p(\mathbf{x}|\mathbf{z})\right] + \mathbb{E}_q\left[\log p(\mathbf{x})\right] \\ = D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{z})||p(\mathbf{z})] - \mathbb{E}_q\left[\log p(\mathbf{x}|\mathbf{z})\right] + \log p(\mathbf{x}). \end{aligned} \tag{1.3}$$

Here, we do not know the normalizing constant of the exact posterior $p(\mathbf{x})$, but since this term does not depend on $\mathbf{z}$, we may ignore it for the purpose of optimizing our approximation q. We are then left with what is called the negative Evidence Lower Bound (negative ELBO):

$$L_q = D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{z})||p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})]. \tag{1.4}$$

We observe that the first term is a regularization that encourages the variational posterior to be close to the prior. On the other hand, the second term is an expected likelihood which encourages variational posteriors that explain the data with high likelihood.

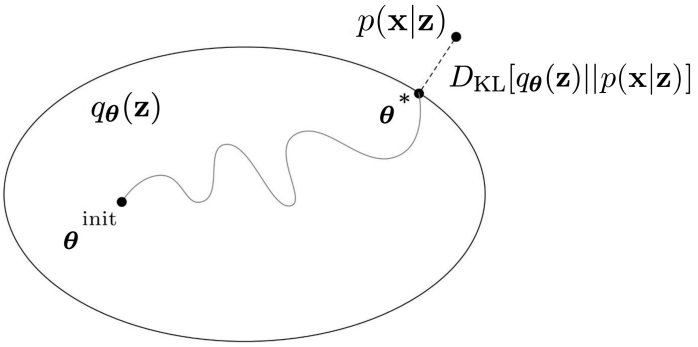


Figure 1.1: Variational Inference as an optimization problem. Variational family approximate posterior $q_{\boldsymbol{\theta}}(\mathbf{z})$ is optimised to minimise D_{KL} from true posterior $p(\mathbf{z}|\mathbf{x})$. Source: Blei et al. (2016).

In practice, the expectation of the log-likelihood $p(\mathbf{x}|\mathbf{z})$ with respect to q is usually not analytically tractable and instead is estimated using Monte Carlo sampling:

$$\begin{aligned} \mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})] \approx \frac{1}{S}\sum_{s=1}^{S} \log p(\mathbf{x}|\mathbf{z}^{(s)}), \\ \mathbf{z}^{(s)} \sim q_{\boldsymbol{\theta}}(\mathbf{z}), \end{aligned} \tag{1.5}$$

where the ELBO is optimized by differentiating this stochastic approximation with respect to the variational parameters $\boldsymbol{\theta}$ (Salimans et al., 2013; Kingma & Welling, 2013). By optimizing the parameters of the variational distribution, we iteratively refine the approximation until it becomes close to the true posterior. See Figure 1.1. This process is often performed using stochastic gradient descent or other optimization techniques.

We can observe that ELBO is a lower-bound on the log evidence $\log p(\mathbf{x}) \geq \text{ELBO}$ for any $q_{\boldsymbol{\theta}}(\mathbf{z})$, which explains its name. This follows from:

$$D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{z})||p(\mathbf{z}|\mathbf{x})] + \text{ELBO} = \log p(\mathbf{x}) \tag{1.6}$$

and the fact that $\mathrm{KL}(\cdot) \geq 0$ (Kullback & Leibler, 1951).

Finally, in this thesis, we study the setting of *amortized variational inference* (Gershman & Goodman, 2014) where the variational parameters are shared across data points.

1.5.1 Gaussian Mean-Field Variational Inference

Gaussian Mean-Field Variational Inference (GMFVI) (Blei et al., 2017; Blundell et al., 2015b) is a widely used approach within variational inference that assumes the approximate posterior distribution can be factorized into independent Gaussian distributions $q = \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ with $z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. This mean-field assumption simplifies the problem by assuming that the latent variables are not correlated, allowing for more efficient computations. The latents z_i

can then be sampled using a reparameterization trick (Kingma & Welling, 2013), i.e, for the s-th sample, we have

$$z_i^{(s)} = \mu_i + \sigma_i \epsilon^{(s)}, \qquad \epsilon \sim \mathcal{N}(0, 1). \tag{1.7}$$

This makes the parameters μ_i and σ_i differentiable.

Variational inference provides several advantages, particularly in the context of large and complex models. It is computationally efficient compared to other methods, such as Markov chain Monte Carlo (MCMC), making it scalable to large datasets and complex models. Moreover, variational inference allows for parallel computation and can scale to massive datasets, enabling real-time or online inference, and generally provide competitive performance (Ovadia et al., 2019a).

However, variational inference relies on approximations, and the quality of the approximation depends on the chosen variational family. The chosen family may introduce biases or inaccuracies in the inference process. Additionally, variational inference can struggle with capturing complex dependencies and multimodal posterior distributions, leading to less accurate results compared to exact Bayesian inference methods (Giordano et al., 2018).

Despite these limitations, variational inference remains a valuable tool for Bayesian inference in scenarios where exact calculations are impractical or computationally prohibitive. Its ability to provide fast and scalable approximations has made it a popular choice for a wide range of applications in machine learning, probabilistic modelling, Bayesian statistics, and deep learning.

Chapter 2

Variational Inference in Bayesian Neural Networks

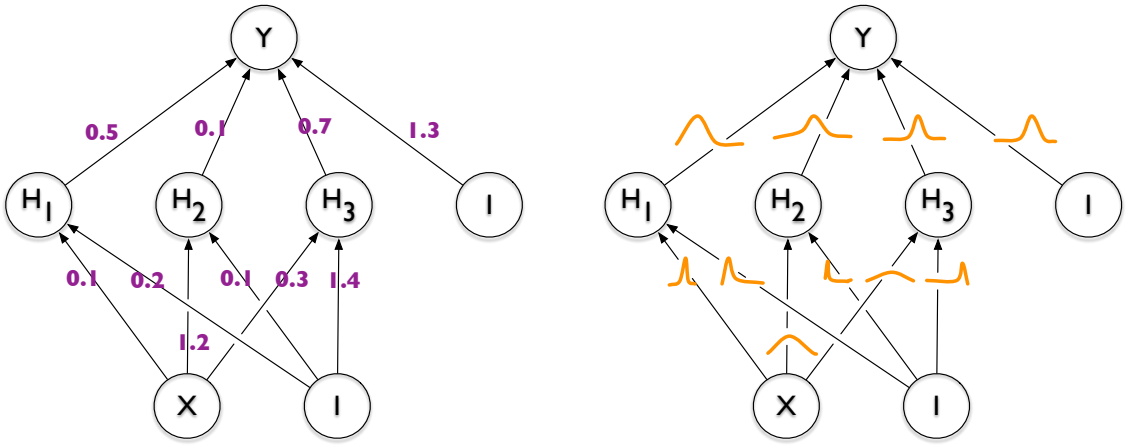
Neural networks have been widely successful in various domains, demonstrating their ability to learn complex patterns and make accurate predictions (Goodfellow et al., 2016). However, traditional neural networks lack a principled approach to quantify uncertainty in their weights and predictions, limiting their reliability in decision-making under uncertainty and robustness analysis (Kendall & Gal, 2017). Bayesian neural networks (BNNs) (MacKay, 1992; Neal, 1993) offer a solution to this problem by integrating Bayesian inference into the learning process, allowing for uncertainty estimation and probabilistic modelling.

2.1 Bayesian Neural Networks

A Bayesian neural network(MacKay, 1992; Neal, 1993) is a type of neural network that leverages the principles of Bayesian probability theory (Pearl, 1988). This offers a robust framework for managing uncertainty, enabling these neural networks to present predictions with a measurable degree of confidence (Gal, 2016). This Bayesian-based approach to neural networks is part of a machine learning subfield known as Bayesian deep learning.

Standard neural networks (Goodfellow et al., 2016) adjust their parameters, such as weights and biases, during training to minimize the loss function. However, they provide deterministic outputs for a given input, meaning if the same input is repeatedly introduced, the output remains unchanged. These networks do not communicate any uncertainty about the predictions, which can be a significant shortcoming in real-world applications where understanding model uncertainty is essential (Kendall & Gal, 2017).

Contrarily, Bayesian neural networks integrate uncertainty within the model parameters $\mathbf{w}$ by treating them as latent variables ($\mathbf{z}$ in Chapter 1). In other words, the weights and biases are interpreted as posterior probability distributions, not single values. See Figure 2.1. This approach allows the network to account for uncertainty in its parameters, leading to uncertainty in predictions. Hence, the outputs from a Bayesian neural network for a given input are not deterministic but probabilistic.



(a) Standard neural network: each weight has a fixed value, as provided by classical backpropagation.

(b) Bayesian neural network: each weight is assigned a distribution, as provided by Bayes by Backprop.

Figure 2.1: Illustrative comparison between a standard neural network and a Bayesian neural network. Source: Blundell et al. (2015a).

For neural network weights $\mathbf{w}$, features $\mathbf{x}$ and labels $\mathbf{y}$, the posterior distribution $p(\mathbf{w}|\mathbf{x},\mathbf{y})$ is computed using Bayes' rule, which multiplies the prior distribution $p(\mathbf{w})$ and data likelihood $p(\mathbf{y}|\mathbf{w},\mathbf{x})$ and renormalizes:

$$p(\mathbf{w}|\mathbf{x},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w},\mathbf{x})p(\mathbf{w})}{p(\mathbf{x},\mathbf{y})}. \tag{2.1}$$

Notice, that while in Chapter 1 we used $\mathbf{x}$ notation for the model outputs as is standard in generative modelling literature, in the Bayesian neural network setting we denote model outputs as $\mathbf{y}$ instead and we use $\mathbf{x}$ as the model conditioning inputs, as is standard in the Bayesian neural network literature.

When predicting with Bayesian neural networks, we form an average over model predictions where each prediction is generated using a set of parameters that is randomly sampled from the posterior distribution:

$$p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{x},\mathbf{y}) = \int p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{w})p(\mathbf{w}|\mathbf{x},\mathbf{y})\,d\mathbf{w}. \tag{2.2}$$

This can be viewed as a type of *ensembling*, of which various types have proven highly effective in deep learning (see e.g. Goodfellow et al., 2016, sec 7.11).

The Bayesian approach offers advantages in several contexts. It can help prevent overfitting to the training data, a common problem in machine learning where a model learns the training data too well, limiting its generalization capabilities (Hinton & Van Camp, 1993a). By considering model parameters as distributions instead of point estimates, Bayesian neural

networks can better regularize their learning, mitigating overfitting (Neal, 1995). Besides offering improved predictive performance over single models, Bayesian ensembles are also more robust because ensemble members will tend to make different predictions on hard examples (Raftery et al., 2005). The diversity of the ensemble represents predictive uncertainty and can be used for out-of-domain detection or other risk-sensitive applications (Ovadia et al., 2019a).

Despite these advantages, Bayesian neural networks come with their own set of difficulties. The most notable is the computational cost associated with estimating the posterior distribution over the model parameters, especially for large networks (Graves, 2011a). Various approximations, such as the Variational Inference Graves (2011b); Ranganath et al. (2014); Blundell et al. (2015b); Hernández-Lobato & Adams (2015); Zhang et al. (2017); Khan et al. (2018) applied in P1 and the Stochastic Gradient Markov Chain Monte Carlo (Welling & Teh, 2011; Chen et al., 2014) applied in P2, have been developed to handle this issue. We describe both types of approximations in subsequent sections.

2.2 Variational Inference Bayesian Neural Networks

Variational inference, described in Section 1.5, is the most widespread approach for training Bayesian neural networks. It approximates the intractable true posterior distribution over model parameters $p(\mathbf{w}|\mathbf{x},\mathbf{y})$ using a tractable simpler distribution $q_{\boldsymbol{\theta}}(\mathbf{w})$. This tractable distribution is parameterised by variational parameters θ that are optimized to minimize the distance between the two distributions:

$$\begin{aligned} D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{w})||p(\mathbf{w}|\mathbf{x},\mathbf{y})] &= \mathbb{E}_q\left[\log \frac{q_{\boldsymbol{\theta}}(\mathbf{w})}{p(\mathbf{w}|\mathbf{x},\mathbf{y})}\right] \\ &= \mathbb{E}_q\left[\log \frac{q_{\boldsymbol{\theta}}(\mathbf{w})}{p(\mathbf{w})p(\mathbf{y}|\mathbf{w},\mathbf{x})/p(\mathbf{x},\mathbf{y})}\right]. \end{aligned} \tag{2.3}$$

While the true posterior distribution is unknown due to the intractable marginal evidence $p(\mathbf{x},\mathbf{y})$, we can still optimise the above distance with respect to $\boldsymbol{\theta}$ because the evidence $p(\mathbf{x},\mathbf{y})$ does not depend on $\boldsymbol{\theta}$. This results in the negative ELBO loss function:

$$L_q = D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{w})||p(\mathbf{w})] - \mathbb{E}_q[\log p(\mathbf{y}|\mathbf{w},\mathbf{x})]. \tag{2.4}$$

were the expectation over the log-likelihood is usually approximated using Monte Carlo sampling:

$$\begin{aligned} \mathbb{E}_q[\log p(\mathbf{y}|\mathbf{w},\mathbf{x})] \approx \frac{1}{S}\sum_{s=1}^{S} \log p(\mathbf{y}|\mathbf{w}^{(s)},\mathbf{x}), \\ \mathbf{w}^{(s)} \sim q_{\boldsymbol{\theta}}(\mathbf{w}), \end{aligned} \tag{2.5}$$

In practice, it is common to use only a single parameter sample $\mathbf{w}^{(s)}$ for this approximation during each training iteration of variational inference Bayesian neural networks.

Given trained $q_{\boldsymbol{\theta}}(\mathbf{w})$ we *predict* on a new instance x^* by integrating out the model parameters as shown in Equation 2.2. However, the exact integration is not tractable. Therefore, we approximate it using Monte Carlo sampling from $q_{\boldsymbol{\theta}}(\mathbf{w})$:

$$p(y^*|x^*, \mathbf{x}, \mathbf{y}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y^*|\mathbf{w}^{(s)}, x^*), \tag{2.6}$$

where $\mathbf{w}^{(s)}$, $s = 1, \ldots, S$, is sampled from $q_{\boldsymbol{\theta}}(\mathbf{w})$.

In Bayesian neural networks, we consider layers that consist of a linear transformation followed by a non-linearity f,

$$\mathbf{a}_l = \mathbf{h}_l \mathbf{W}_l + \mathbf{b}_l, \qquad \mathbf{h}_{l+1} = f(\mathbf{a}_l), \tag{2.7}$$

where $\mathbf{W}_l \in \mathbb{R}^{m \times n}$, $\mathbf{h}_l \in \mathbb{R}^{1 \times m}$ and $\mathbf{b}_l, \mathbf{a}_l, \mathbf{h}_{l+1} \in \mathbb{R}^{1 \times n}$. To simplify the notation in the following, we drop the subscript l such that $\mathbf{W} = \mathbf{W}_l$, $\boldsymbol{\mu}_q = \boldsymbol{\mu}_{ql}$, $\boldsymbol{\Sigma}_q = \boldsymbol{\Sigma}_{ql}$ and we focus on the kernel matrix $\mathbf{W}$ for a single layer.

2.2.1 Gaussian Mean-Field Variatonal Inference Bayesian Neural Network

The most widely adopted parameterization for training the variational inference Bayesian neural networks is Gaussian Mean-Field Variational Inference (GMFVI)(Blundell et al., 2015a), introduced more generally in Section 1.5.1. In GMFVI, we model the variational posterior as:

$$\begin{aligned} q(\mathbf{W}) = \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) = \prod_{i=1}^{m} \prod_{j=1}^{n} q(w_{ij}), \\ \text{with} \quad q(w_{ij}) = \mathcal{N}(\mu_{ij}, \sigma_{ij}^2), \end{aligned} \tag{2.8}$$

where $\boldsymbol{\mu}_q \in \mathbb{R}^{mn \times 1}$ is the posterior mean vector, $\boldsymbol{\Sigma}_q \in \mathbb{R}_+^{mn \times mn}$ is the diagonal posterior covariance matrix.

The weights are then usually sampled using a reparameterization trick (Kingma & Welling, 2013), i.e, for the s-th sample, we have

$$w_{ij}^{(s)} = \mu_{ij} + \sigma_{ij} \epsilon^{(s)}, \qquad \epsilon \sim \mathcal{N}(0, 1). \tag{2.9}$$

In practice, we often represent the posterior standard deviation parameters σ_{ij} in the form of a matrix $\mathbf{A} \in \mathbb{R}_+^{m \times n}$. Note that we have the relationship $\boldsymbol{\Sigma}_q = \text{diag}(\text{vec}(\mathbf{A}^2))$ where the elementwise-squared $\mathbf{A}$ is vectorized by stacking its columns, and then expanded as a diagonal matrix into $\mathbb{R}_+^{mn \times mn}$. See Figure 2.2 for illustration.

While Gaussian Mean-Field posteriors are considered to be one of the simplest types of variational approximations, with some known limitations (Giordano et al., 2018), they scale to comparatively large models and generally provide competitive performance (Ovadia et al., 2019a). Additionally, Farquhar et al. (2020b) have found that the Mean-Field becomes a

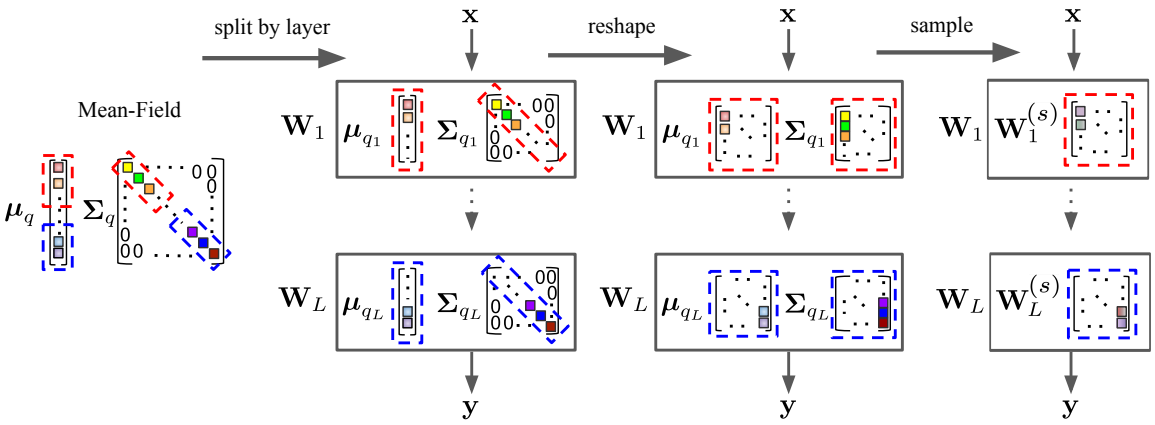


Figure 2.2: Illustration of sampling a neural network ensemble member from a GMFVI Bayesian neural network.

less restrictive assumption as the depth of the network increases. However, when compared to deterministic neural networks, GMFVI doubles the number of parameters and is often harder to train due to the increased noise in stochastic gradient estimates. We address these challenges in Chapter 4 of this thesis. Furthermore, despite the theoretical advantages of GMFVI over the deterministic neural networks, GMFVI suffers from over-regularization for larger networks, which leads to underfitting and often worse predictive performance in such settings, as we show in Chapter 5.

2.3 SG-MCMC Bayesian Neural Networks

Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC) methods are an alternative to variational inference for approximation of the parameter posterior distribution in Bayesian neural networks. SC-MCMC Bayesian neural networks are the main topic of Chapter 5. SG-MCMC methods approximate the posterior distribution by combining stochastic gradient descent with Markov chain Monte Carlo techniques, allowing for scalable and approximate sampling from the posterior. These methods generate samples by iteratively updating the weights using stochastic gradients and applying Markov chain transitions to explore the posterior distribution. Unlike the variational inference approximation, the samples from the posterior should in the limit converge to the true posterior distribution. In this SG-MCMC setting, we denote the neural network weights $\mathbf{w}$ as $\boldsymbol{\theta}$ because they are the object of optimisation (unlike in variational inference where the variational parameters are optimised).

In supervised deep learning we use a training dataset of features $\mathbf{x} = \{x_i\}_{i=1,\dots,n}$ and labels $\mathbf{y} = \{y_i\}_{i=1,\dots,n}$, and a probabilistic model $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$ to minimize the regularized

cross-entropy objective,

$$L(\boldsymbol{\theta}) := -\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta}) + \Omega(\boldsymbol{\theta}), \tag{2.10}$$

where $\Omega(\boldsymbol{\theta})$ is a regularizer over model parameters. We approximately optimize (2.10) using variants of stochastic gradient descent (SGD), (Sutskever et al., 2013). Beside being efficient, the SGD minibatch noise also has generalization benefits (Masters & Luschi, 2018; Mandt et al., 2017).

SG-MCMC methods reframe the standard deep learning parameter optimization as sampling from posterior distribution over the parameters:

$$p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{x})p(\boldsymbol{\theta}) = \exp(-U(\boldsymbol{\theta})/T), \tag{2.11}$$

where $U(\boldsymbol{\theta})$ is the *posterior energy function* analogous to a standard supervised learning training objective (2.10):

$$U(\boldsymbol{\theta}) := -\log p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{x}) - \log p(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}), \tag{2.12}$$

where the first term is the likelihood of data and the second term is regularization. If we scale $U(\boldsymbol{\theta})$ by $1/n$ and set $\Omega(\boldsymbol{\theta}) = -\frac{1}{n}\log p(\boldsymbol{\theta})$ we recover $L(\boldsymbol{\theta})$ in (2.10). Therefore $\exp(-U(\boldsymbol{\theta}))$ simply gives high probability to models which have low loss $L(\boldsymbol{\theta})$. T is a *temperature* that adjusts the sharpness of the posterior distribution. Approximation of the posterior with sampling proportional to the likelihood and prior avoids the computation of the intractable normalizing term in the exact Bayes posterior equation.

Given $p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y})$ we *predict* on a new instance x^* by averaging over all likely models as show in Equation 2.2. However, solving this integral exactly is not possible. Instead, we approximate the integral using Monte Carlo sampling:

$$p(y^*|x^*, \mathbf{x}, \mathbf{y}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y^*|x^*, \boldsymbol{\theta}^{(s)}), \tag{2.13}$$

where $\boldsymbol{\theta}^{(s)}$, $s = 1, \ldots, S$, are samples saved during the Markov chain transitions when exploring the posterior distribution $p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y})$. This is in contrast to the Variational Inference BNNs that generate the posterior samples at prediction time.

2.3.1 Posterior Simulation using Langevin Dynamics

To generate approximate parameter samples $\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \,|\, \mathcal{D})$ we consider *Langevin dynamics* over parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ and momenta $\mathbf{m} \in \mathbb{R}^d$, defined by the Langevin stochastic differential equation (SDE),

$$\mathrm{d}\,\boldsymbol{\theta} = \mathbf{M}^{-1}\,\mathbf{m}\,\mathrm{d}t, \tag{2.14}$$

$$\mathrm{d}\,\mathbf{m} = -\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\,\mathrm{d}t - \gamma\mathbf{m}\,\mathrm{d}t + \sqrt{2\gamma T}\,\mathbf{M}^{1/2}\,\mathrm{d}\mathbf{W}. \tag{2.15}$$

Here $U(\boldsymbol{\theta})$ is the *posterior energy* defined in (2.12), and $T > 0$ is the *temperature*. We use $\mathbf{W}$ to denote a standard multivariate Wiener process, which we can loosely understand as a generalized Gaussian distribution (Särkkä & Solin, 2019; Leimkuhler & Matthews, 2016). The *mass matrix* $\mathbf{M}$ is a preconditioner, and if we use no preconditioner then $\mathbf{M} = I$, such that all $\mathbf{M}$-related terms vanish from the equations. The *friction* parameter $\gamma > 0$ controls both the strength of coupling between the moments $\mathbf{m}$ and parameters $\boldsymbol{\theta}$ as well as the amount of injected noise (Langevin, 1908; Leimkuhler & Matthews, 2016). For any friction $\gamma > 0$ the SDE (2.14–2.15) has the same limiting distribution, but the choice of friction *does* affect the speed of convergence to this distribution. Simulating the continuous Langevin SDE (2.14–2.15) produces a trajectory distributed according to $\exp(-U(\boldsymbol{\theta})/T)$ and the Bayes posterior is recovered for $T = 1$.

2.3.2 Stochastic Gradient MCMC (SG-MCMC)

Bayesian inference now corresponds to simulating the above SDE (2.14–2.15) and this requires numerical discretization. For efficiency *stochastic gradient Markov chain Monte Carlo* (SG-MCMC) methods further approximate $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ with a minibatch gradient (Welling & Teh, 2011; Chen et al., 2014). For a minibatch $B \subset \{1, 2, \ldots, n\}$ we first compute the minibatch average gradient $\tilde{G}(\boldsymbol{\theta})$,

$$\nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}) := -\frac{1}{|B|} \sum_{i \in B} \nabla_{\boldsymbol{\theta}} \log p(y_i | x_i, \boldsymbol{\theta}) - \frac{1}{n} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}), \tag{2.16}$$

and approximate $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ with the unbiased estimate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}) = n \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta})$. Here $|B|$ is the minibatch size and n is the training set size; in particular, note that the log prior scales with $1/n$ regardless of the batch size.

The SDE (2.14–2.15) is defined in continuous time ($\mathrm{d}t$), and in order to solve the dynamics numerically we have to discretize the time domain (Särkkä & Solin, 2019). In this work we use a simple first-order symplectic Euler discretization, (Leimkuhler & Matthews, 2016), as first proposed for (2.14–2.15) by Chen et al. (2014). Recent work has used more sophisticated discretizations, (Chen et al., 2015; Shang et al., 2015; Heber et al., 2019; Heek & Kalchbrenner, 2019). Applying the symplectic Euler scheme to (2.14–2.15) gives the discrete time update equations,

$$\mathbf{m}^{(t)} = (1 - h\gamma)\, \mathbf{m}^{(t-1)} - hn \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)}) \tag{2.17}$$

$$+ \sqrt{2\gamma h T}\, \mathbf{M}^{1/2}\, \mathbf{R}^{(t)}, \tag{2.18}$$

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} + h\, \mathbf{M}^{-1} \mathbf{m}^{(t)}, \tag{2.19}$$

where $\mathbf{R}^{(t)} \sim \mathcal{N}_d(0, I_d)$ is a standard Normal vector.

To put this into perspective, when $\mathbf{M} = I$ the only difference between these updates and the familiar SGD with momentum is the additional noise term $\sqrt{2\gamma h T}\, \mathbf{M}^{1/2} \mathbf{R}^{(t)}$ in the SG-MCMC method. In this *precise* sense the presented SG-MCMC is just "SGD with noise". In fact, the parameterization in terms of step size h and friction γ, can be

rewritten in terms of the familiar SGD learning rate ℓ and momentum decay parameters β by setting $h := \sqrt{\ell/n}$, and $\gamma := (1-\beta)\sqrt{n/\ell}$. Appendix B.2 contains exact derivations of these equations. Furthermore, the preconditioner $\mathbf{M}$ can be viewed as the familiar adaptive learning rate preconditioners known from commonly used optimizers such RMSprop (Tieleman & Hinton, 2012) or Adam (Kingma & Ba, 2014).

In summary, SG-MCMC BNNs approximate the posterior using samples stored from exploring the posterior space, instead of approximating the posterior using a simpler distribution as variational inference BNNs. While variational BNNs avoid computing the marginal evidence by minimising the ELBO objective that is independent of the marginal evidence, SG-MCMC BNNs achieve it by sampling, which also does not require computing the marginal evidence. The variational BNNs samples from the variational distribution during prediction, while the SG-MCMC reuse the samples stored from the exploration. Both the variational inference and SG-MCMC methods inject additional noise in the training procedure compared to deterministic neural networks. In variational inference, the additional noise comes from sampling the network parameters from the variational posterior in each forward pass during training. In SG-MCMC, the additional noise is instead injected during the network parameter updates. Therefore, while the implementation details are different for both of the described BNN approaches, the core principles of sampling ensemble members and injecting additional parameter noise during training are similar.

2.4 Summary of Chapter 4

In this section, we provide a concise summary of the contributions and results of paper P1, which is later described in more detail in Chapter 4.

2.4.1 Introduction

Beyond mean-field variational inference (Section 1.5.1), recent work on approximate Bayesian inference has explored ever richer parameterizations of the approximate posterior in the hope of improving the performance of Bayesian neural networks. In contrast, here we study a simpler, more compactly parameterized variational approximation. Our motivation for studying this setting is to better understand the behaviour of GMFVI with the goal to address the issues with its practical applicability. Consequently, we show that the compact approximations can also work well for a variety of models. In particular we find that:

- Converged posterior standard deviations under GMFVI consistently display strong low-rank structure. This means that by decomposing these variational parameters into a low-rank factorization, we can make our variational approximation more compact without decreasing our model's performance.
- Factorized parameterizations of posterior standard deviations improve the signal-to-noise ratio of stochastic gradient estimates, and thus not only reduce the number of parameters compared to standard GMFVI, but also can lead to faster convergence.

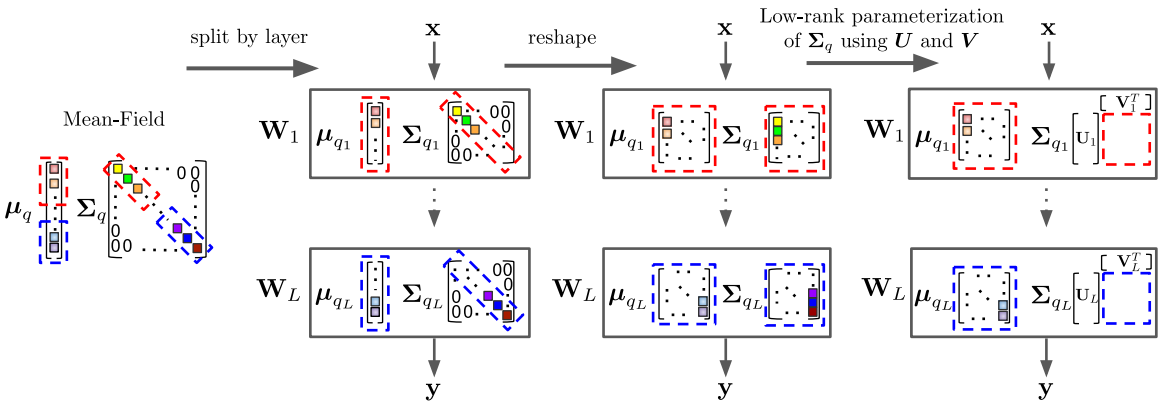


Figure 2.3: Illustration of the relationship between the standard Gaussian Mean-Field posterior and its "low-rank" parameterization, which we call the k-tied Normal posterior. The illustration shows the posterior parameterization for a network with L layers, where $\mathbf{x}$ and $\mathbf{y}$ are the network inputs and outputs respectively, and $\boldsymbol{\mu}_{q_1}$, $\boldsymbol{\Sigma}_{q_1}$, $\boldsymbol{\mu}_{q_L}$ and $\boldsymbol{\Sigma}_{q_L}$ are the variational parameters for the layers 1 and L respectively. The k-tied Normal distribution parameterizes the already diagonal per layer posterior covariance matrices $\boldsymbol{\Sigma}_{q_{1..L}}$ using the even more compact $\boldsymbol{U}_{1..L}$ and $\boldsymbol{V}_{1..L}$ matrices from $\mathcal{N}\big(\boldsymbol{\mu}_q, \mathrm{diag}\big(\mathrm{vec}\big((\mathbf{U}\mathbf{V}^T)^2\big)\big)\big)$.

2.4.2 Results

We start by empirically studying the properties of the spectrum of matrices $\mathbf{A}$, introduced in Section 2.2, *post-training* (after convergence), while using standard Gaussian mean-field variational distributions (see Figure 2.3). Interestingly, we observe that those matrices naturally exhibit a low-rank structure (see Figure 2.4), i.e,

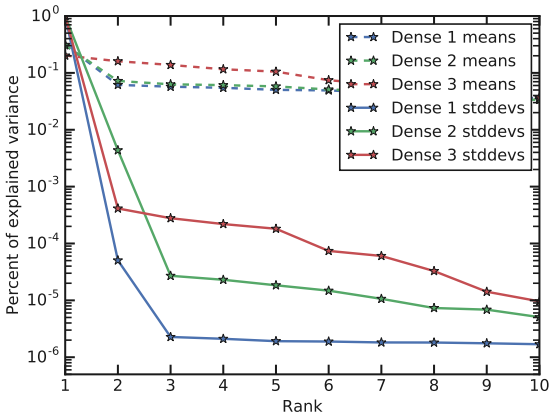
$$\mathbf{A} \approx \mathbf{U}\mathbf{V}^T \tag{2.20}$$

for some $\mathbf{U} \in \mathbb{R}^{m \times k}$, $\mathbf{V} \in \mathbb{R}^{n \times k}$ and k a small value (e.g., 2 or 3). This observation motivates the introduction of the following variational family, which we name k-tied Normal:

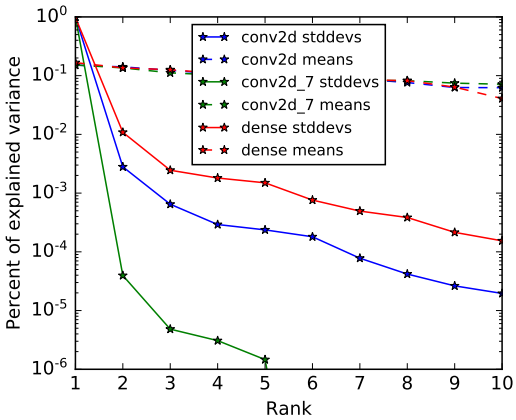
$$\boxed{\begin{aligned} &k\text{-}tied\text{-}\mathcal{N}(\mathbf{W}; \boldsymbol{\mu}_q, \mathbf{U}, \mathbf{V}) = \\ &\mathcal{N}\big(\boldsymbol{\mu}_q, \mathrm{diag}\big(\mathrm{vec}\big((\mathbf{U}\mathbf{V}^T)^2\big)\big)\big), \end{aligned}} \tag{2.21}$$

where the squaring of the matrix $\mathbf{U}\mathbf{V}^T$ is applied elementwise. Due to the tied parameterization of the diagonal covariance matrix, we emphasize that this variational family is *smaller*—i.e., included in—the standard Gaussian mean-field variational distribution family.

Notice that our diagonal covariance $\boldsymbol{\Sigma}_q$ repeatedly reuses the same elements of $\mathbf{U}$ and $\mathbf{V}$, which results in parameter sharing across different weights. The total number of the standard deviation parameters in our method is $k(m+n)$ from $\mathbf{U}$ and $\mathbf{V}$, compared to mn from $\mathbf{A}$ in the standard GMFVI parameterization. Given that in our experiments, the k is very low (e.g. $k = 2$), this reduces the number of the posterior standard deviation parameters from quadratic to linear in the dimensions of the layer. This in practice approximately halves

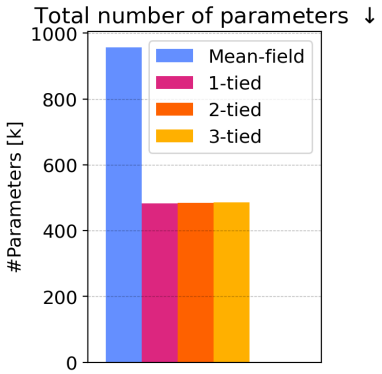


(a) 3-layer MLP trained on MNIST

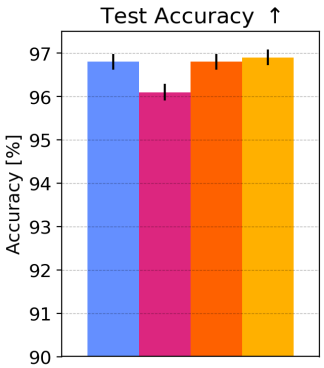


(b) ResNet-18 trained on CIFAR-10

Figure 2.4: Fraction of variance explained per each singular value from SVD of matrices of posterior means and posterior standard deviations post-training in different model types. Unlike posterior means, posterior standard deviations clearly display a strong low-rank structure, with most of the variance contained in the top few singular values.



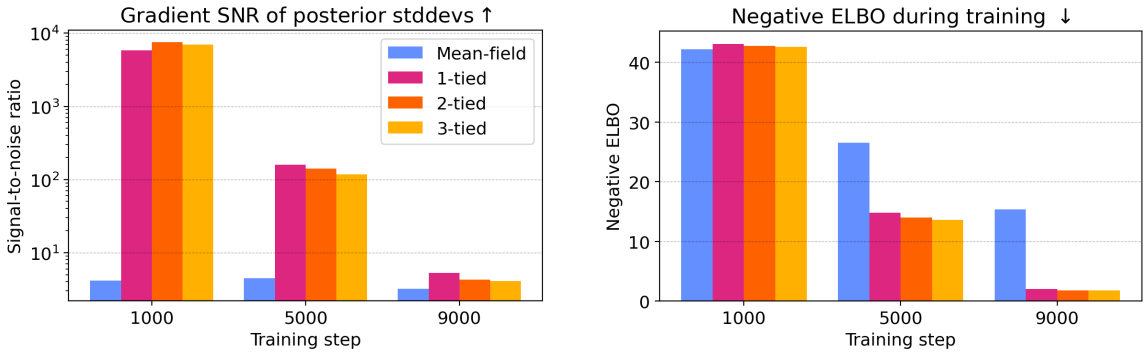
(a) The k-tied Normal distribution approximately halves the number of model parameters.



(b) The k-tied Normal distribution does not degrade predictive performance for $k \geq 2$.

Figure 2.5: The k-tied Normal distribution with rank $k \geq 2$ approximately halves the number of model parameters without decreasing the predictive performance compared to the standard GMFVI.

the total number of model parameters (posterior means are unchanged) without decreasing predictive performance (see Figure 2.5). Additionally, such parameter sharing across the weights leads to higher signal-to-noise ratio during training and thus in some cases faster convergence (see Figure 2.6).



(a) Mean gradient Signal-to-Noise-Ratio (SNR) in the log posterior standard deviation parameters of the MNIST MLP model at increasing training steps for different ranks of tying k. The k-tied Normal distribution significantly increases the SNR for these parameters.

(b) Negative ELBO on the MNIST validation dataset at increasing training steps for different ranks of tying k. The higher SNR from the k-tied Normal posterior translates into the increased convergence speed for the MLP model.

Figure 2.6: The k-tied Normal distribution improves the signal-to-noise ratio of stochastic gradient estimates and results in faster training convergence speed.

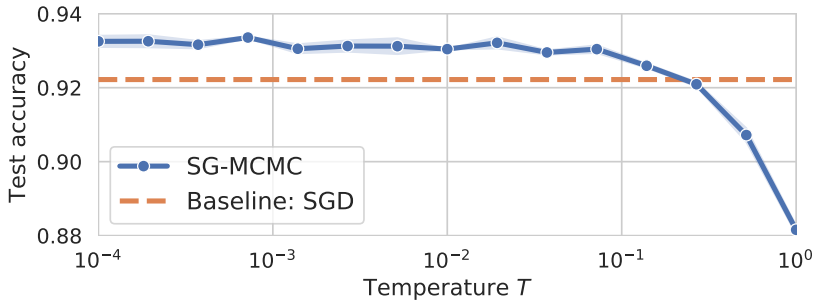


Figure 2.7: The "**cold posterior**" effect: for a ResNet-20 on CIFAR-10 we can improve the generalization performance significantly by cooling the posterior with a temperature $T \ll 1$, deviating from the Bayes posterior $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-U(\boldsymbol{\theta})/T)$ at $T = 1$.

2.5 Summary of Chapter 5

This section summarizes the contributions of Paper P2, which is described in greater detail in Chapter 5.

Paper P2 reevaluates the understanding of Bayes posteriors in popular deep neural networks. It shows that theoretically optimal Bayes posteriors yield systematically worse predictions compared to standard SGD optimisation and posteriors with an artificially downscaled temperature $T < 1$ in Equation 2.11, as illustrated in Figure 2.7. This phenomenon is termed the "cold posteriors" effect, where the temperature $T < 1$ corresponds to overcounting the data evidence by a factor of $1/T$. Paper P2 also highlights the prevalence of such downscaling in the Bayesian neural network literature.

Additionally, P2 proposes several hypotheses that could explain the cold posterior effect, and these hypotheses are evaluated through experimental analysis. The hypotheses are grouped based on their connection with either the SG-MCMC inference procedure, the prior, or the likelihood function.

First, the hypothesis related to the SG-MCMC inference procedure suggests that this procedure may be inaccurate. The high-dimensional posterior of deep neural networks could lead to difficult-to-simulate Stochastic Differential Equation (SDE) dynamics (Equations 2.14-2.15). Moreover, the approximate SG-MCMC inference method has to contend with minibatch noise and it produces only a finite sample approximation to the predictive integral in Equation 2.13. However, P2 shows that the SG-MCMC inference procedure is accurate in the analysed settings, indicating that it is not the cause of the cold posterior effect.

Second, P2 investigates the effects of deep learning practices that violate the likelihood principle, such as batch normalization, dropout, and data augmentation. It discovers that the cold posterior effect persists even when a clean likelihood function is employed.

Finally, P2 provides evidence suggesting that the standard Normal prior commonly used for Bayesian inference is inadequate for Bayesian neural networks. Specifically, it shows that typical functions produced by this prior assign a high probability to the same class for all inputs, whereas a uniform distribution would be expected. This implicates the prior in the cold posterior effect.

Chapter 3

Variational Inference in Neural Speech Synthesis

3.1 Overview

This chapter is centred around the use of Variational Inference in the realm of Neural Speech Synthesis (Tan, 2023). Here, we examine and explore the conjunction of deep learning techniques and digital audio processing in synthesizing human speech. With a focus on Variational Auto-Encoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014), we will delve into their application and significance in creating more refined and human-like synthesized speech.

To set the foundation for the discussion, the chapter starts with a comprehensive examination of speech synthesis in section 3.2. It begins with the task formulation of speech synthesis, outlining the problem space and associated challenges. Next, it delves into the technical aspects of speech signal processing. We then examine the high-level system architecture typical of a speech synthesis system (Taylor, 2009).

To understand the building blocks of speech synthesis systems, individual sub-components like text analysis, acoustic models, and vocoders are dissected and analyzed. We then describe a recent trend in combining the acoustic and vocoder submodules with end-to-end modelling. This will enable an understanding of the fundamental techniques and architectures utilized in speech synthesis, setting the stage for later discussions on applying variational inference in such systems.

Section 3.3 will begin with an introduction to the Variational Auto-Encoder, providing the theoretical background needed to understand how it works, its architecture, and how it differs from standard auto-encoders (Hinton & Salakhutdinov, 2006; Goodfellow et al., 2016). We will describe how the principles of variational inference introduced in Section 1.5 are applied in the context of the VAE. This section will establish the foundation for understanding the relevance of VAEs in the speech synthesis process.

Section 3.4, "VAE in Speech Synthesis," dives deeper into the use of VAEs in the speech synthesis process. We will discuss the role of VAEs in enhancing the quality and versatility

of synthesized speech. We further subdivide this section into two sub-sections: 3.4.1, "VAE for Prosody Transfer," and 3.4.2, "VAE for Audio Representation Learning." In these sub-sections, we will discuss how VAEs can be used to capture and transfer prosody from one speech to another and for learning compact, meaningful representations of audio data. Prosody, which refers to the rhythm, stress, and intonation of speech, is a crucial aspect of natural-sounding synthesized speech, and VAEs provide a promising approach to capturing and transferring this aspect of speech.

Section 3.5, "Machine Dubbing," moves us into an application area of the principles discussed earlier. Machine dubbing signifies the replacement of original voice content with synthesized voices in different languages while maintaining the original emotion and tone. We will look at how the concepts of VAE and speech synthesis come together in this practical and growing field.

Finally, Sections 3.6 and 3.7 summarise two machine dubbing papers P3 and P4 which are the contributions of this thesis. P3 proposes a cross-lingual prosody transfer method based on VAE for machine dubbing. P4 extends this method with a more granular prosody modelling using the VAE representations, which improves the quality of machine dubs.

In essence, this chapter aims to give an in-depth view of the application of variational inference in neural speech synthesis, its potential in machine dubbing, and its growing relevance in today's world of AI and machine learning. This work aspires to bridge the gap between theoretical understanding and practical implementations, providing insights that can be utilized for further research and development.

3.2 Speech Synthesis

3.2.1 Task formulation

Speech synthesis, also often referred to as text-to-speech (TTS), is a field of study within artificial intelligence and computational linguistics that focuses on generating human-like speech from text and other data. More formally, given an input text sequence, the goal of a speech synthesis system is to generate a corresponding audio waveform that a human listener would perceive as natural speech. See Figure 3.1. This process involves mapping linguistic symbols into a sequence of acoustic parameters, and subsequently converting these parameters into a digital audio signal. This task is typically framed as a sequence-to-sequence problem, where the input sequence is the text and the output sequence is the audio waveform. Speech synthesis has broad applications, ranging from accessibility features for the visually impaired or dyslexic individuals to its use in digital personal assistants, navigation systems, call centres, audiobook reading and more recently in entertainment such as video games and video dubbing.

An important aspect of the task formulation is determining what constitutes 'natural speech'. It is not enough for the system to simply articulate words—it must also recreate the intonations, pauses, and other prosodic features that occur in human speech, and do so in a way that matches the context of the text being spoken. The goal, therefore, is not just

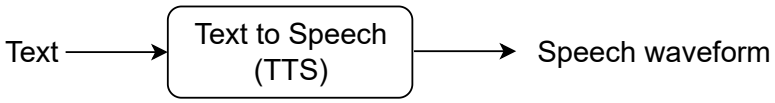


Figure 3.1: Conventional Text-To-Speech systems generate speech waveform for an input text.

intelligibility, but also naturalness and expressivity.

Furthermore, real-world applications often require control over aspects of the speech such as the speaker's identity, accent, emotional state, speaking style, and acoustic condition. This implies an additional layer of complexity in the task formulation, as the system must learn to control these aspects in a way that is both consistent with the input text and flexible to the specific requirements of the application. Conventional text-to-speech systems as shown in Figure 3.1 often do not meet these requirements. We will delve into the controllability of speech synthesis in Section 3.2.7 and focus on its application in the challenging machine dubbing domain in Section 3.5. Preceding sections will assume a single-speaker, single-language, neutral text-to-speech system trained on studio quality recording.

3.2.2 Speech signal processing

This section describes concepts instrumental to the field of speech signal processing, thereby fortifying the understanding required for subsequent discussions.

Analog-to-Digital Conversion

Analog-to-Digital Conversion (ADC) plays a fundamental role in the field of digital signal processing and, by extension, in speech synthesis. While human speech is an inherently analogue phenomenon—continuous in both time and amplitude—it is necessary to convert it into a digital format to process it using computer systems. This conversion is accomplished through ADC. The process of ADC consists of two primary steps: sampling and quantization.

Sampling involves recording the amplitude of the speech signal at discrete intervals. The sampling rate, or the number of samples taken per second, determines how accurately the digital signal can represent the analogue signal. In the realm of speech synthesis, a common sampling rate is 24 kHz.

Quantization is the process of assigning discrete amplitude values to each sample. It essentially involves converting the continuous range of amplitudes in the analogue signal to a finite set of possible amplitude levels. The number of levels is determined by the bit depth of the quantization, with a higher bit depth providing a more accurate representation of the original amplitudes, but also requiring more memory to store. It is common to use 16-bit per sample. Higher bit rates would increase memory consumption without high benefits to perceptual quality.

The result of ADC is a digital audio signal, represented as a sequence of discrete samples, each with a specific amplitude value. This digital signal serves as the basis for further processing.

Time to Frequency Domain Transformation

Once the speech signal has been digitized through the analogue-to-digital conversion, the next critical step in speech signal processing involves transforming the signal from the time domain to the frequency domain. This transformation provides a different representation of the signal, revealing the different frequencies that make up the signal and their relative intensities.

The transformation from the time domain to the frequency domain is typically achieved through a mathematical operation called the Fourier Transform (Bracewell & Bracewell, 1986). In practice, for digital signals, a variant of the Fourier Transform known as the Fast Fourier Transform (FFT) is commonly used due to its computational efficiency. The FFT essentially decomposes the time-domain signal into a sum of sinusoidal components of different frequencies, each with its own amplitude and phase. The result is a spectrum that shows the relative intensity of each frequency component in the signal.

Inherent to human speech is a non-stationary nature of the speech signal. Consequently, the static Fourier Transform, which provides a global view of the frequency components, becomes inadequate for capturing this time-varying behaviour. To address this, the Short-Time Fourier Transform (STFT) (Allen, 1977) is typically employed.

The STFT divides the continuous speech signal into small, overlapping segments or frames using a window function, and then applies the Fourier Transform to each segment. This approach allows the spectral characteristics of the speech signal to be analyzed over short time periods, thereby capturing its non-stationary nature.

The choice of the window function and its length have significant implications for the resulting frequency representation. Common window functions include the Hamming window and the Hanning window (Blackman & Tukey, 1958), which mitigate the artificial discontinuities at the edges of each frame, thereby reducing spectral leakage—a distortion in the frequency representation.

The length of the window dictates the time-frequency resolution trade-off. A longer window provides better frequency resolution (i.e., it can distinguish between closely spaced frequencies) but poorer time resolution (i.e., it cannot accurately capture rapid changes in the signal). Conversely, a shorter window offers better time resolution but poorer frequency resolution.

The hop length, or the step size between consecutive windows, also plays a role in this transformation process. Smaller hop sizes lead to a more densely sampled time-frequency representation and hence a lower level of compression, while larger hop sizes yield a more compressed but potentially less detailed representation.

Finally, human perception of sound follows a logarithmic scale and is not linear, and as such, the raw spectrum produced by the STFT does not align well with human auditory perception. Specifically, we are more sensitive to changes in lower frequencies than in higher ones. Mel-spectrograms address this by utilizing the Mel scale (Rabiner & Schafer, 2010), a perceptual scale that approximates the human ear's response to different frequencies. The Mel scale is quasi-logarithmic: it is more densely packed at lower frequencies (which we are more sensitive to) and less densely packed at higher frequencies (which we are less sensitive

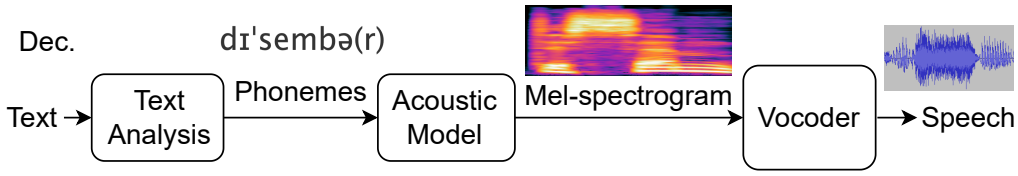


Figure 3.2: Popular multi-stage text-to-speech system architecture.

to).

Equipped with the basic knowledge of speech signal processing, we proceed to a high-level description of speech synthesis systems in the next subsection.

3.2.3 High-level system architecture

The evolution of speech synthesis system architecture reflects the advancements in both computational power and machine learning techniques. Historically, the field was dominated by two main methodologies: Concatenative Synthesis (Hunt & Black, 1996) and Statistical Parametric Synthesis (Zen et al., 2009).

Concatenative Synthesis relied on a vast database of pre-recorded speech segments, which were selected and concatenated to form the output speech. This method, while capable of producing high-quality speech, was limited by the size and diversity of the available speech database.

On the other hand, Statistical Parametric Synthesis used statistical models to generate speech, which was typically more flexible and customizable. However, the speech produced by these methods often sounded more robotic and less natural due to the simplifications and assumptions made by the models.

The advent and proliferation of deep learning techniques have led to a new era in speech synthesis, with Neural Synthesis now being the prevalent approach (Tan, 2023). Neural Synthesis employs neural networks to model and generate speech, leading to synthetic speech that can closely match the naturalness and intelligibility of human speech.

A popular high-level system architecture for modern Neural Synthesis involves a multi-stage approach, consisting of three main components: Text Analysis, Acoustic Modeling, and a Vocoder (Figure 3.2).

Text Analysis is the first stage of this architecture. Here, the input text is transformed into a sequence of phonemes, or distinct units of sound that make up speech. This stage often includes other language processing steps such as tokenization, normalization, part-of-speech tagging, and syntactic analysis, which provide the system with a deeper understanding of the input text. Text Analysis is usually rule-based.

Next, the **Acoustic Model** takes the sequence of phonemes and transforms it into an intermediate acoustic representation, commonly a Mel-spectrogram. This transformation is typically achieved through a neural network model, which can learn complex, non-linear relationships between the phonetic input and the acoustic output.

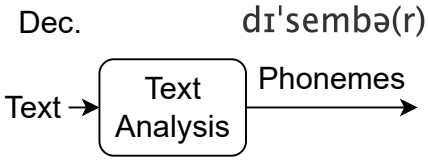


Figure 3.3: Overview of a text analysis module.

The final stage is the **Vocoder**, which transforms the Mel-spectrogram into a speech waveform. This step involves converting the frequency-domain information in the Mel-spectrogram back into the time-domain, resulting in a digitized version of the speech signal. The vocoder is also typically implemented using a neural network, which can generate high-quality, natural-sounding speech.

Subsequent sections provide more detailed descriptions of these three stages.

3.2.4 Text analysis

Text analysis is a crucial component of a speech synthesis system, serving as the first step in the process of transforming written text into spoken speech. It is in this stage that the written input is decoded into a structured representation that exposes the linguistic and phonetic content necessary for generating natural sounding speech.

The text analysis process usually comprises of several sub-stages, including tokenization, text normalization, and phonetic transcription.

Tokenization involves dividing the text into smaller units such as sentences, words, and phrases. This segmentation of text is critical for subsequent stages as it allows the system to analyze and process the text in a more granular and manageable way.

Text normalization is the process of converting written numbers, abbreviations, acronyms, and other non-standard forms of text into their standard spoken equivalents. For instance, "Dr." might be converted to "Doctor," "$20" to "twenty dollars," and "St." to "Street." This stage is crucial to ensure the speech output is intelligible and natural sounding.

Phonetic transcription translates each token into its corresponding phonetic representation. This involves mapping each word to a sequence of phonemes, which are the smallest units of sound in a language. This phonetic sequence serves as the input to the subsequent speech signal processing stages. While neural networks could operate on characters/graphemes directly, the phoneme representation is particularly useful for the controllability of custom pronunciations of words unseen during training, such as entity names.

Text analysis is typically rule-based. The rules are usually developed by linguists that specialize in a given locale, where a locale is composed of a language and an accent. For instance, the English language has accents such as British, United States, Canadian, Australian, Irish, Scottish and Indian.

Text analysis, therefore, involves converting the raw input text into a detailed, structured representation that captures not only the linguistic content of the text, but also the phonetic features that are necessary for generating natural speech. Despite its seeming simplicity, this

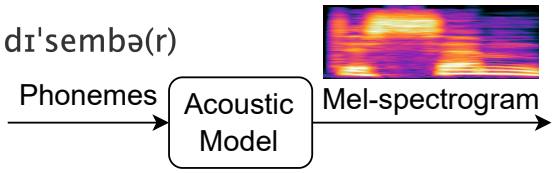


Figure 3.4: Overview of an acoustic module.

.

stage is quite challenging due to the complexity and variability of natural language. It serves as the foundation for the rest of the speech synthesis process, and its quality can significantly impact the naturalness and intelligibility of the synthesized speech.

3.2.5 Acoustic models

Acoustic modelling is a crucial component that plays the role of a sequence-to-sequence converter, transforming sequences of phonemes into corresponding sequences of Mel-spectrogram frames. This transformation task involves establishing a monotonic alignment between the phonemes and the Mel-spectrogram frames. This alignment is essentially the temporal mapping from each phoneme to the segment of the speech waveform it corresponds to, and accurately predicting this alignment is the central problem addressed by the acoustic model.

Two primary approaches have emerged to tackle this alignment problem: auto-regressive modelling and parallel modelling.

Auto-regressive modelling, as embodied in architectures such as Tacotron (Wang et al., 2017; Shen et al., 2018), uses an attention mechanism to predict the alignment between the phonemes and the Mel-spectrogram frames. This approach allows for highly expressive and natural-sounding speech synthesis, as the model can learn complex dependencies between the input and output sequences. However, it tends to be slow due to its sequential nature and can be unstable, sometimes resulting in attention failures where the model loses the correct alignment.

Parallel modelling, in contrast, separates the alignment prediction into a distinct stage based on predicted phoneme durations. This approach, used in models like Parallel Tacotron (Elias et al., 2021), offers faster and more stable synthesis as it decouples duration prediction from the rest of the acoustic model. However, it can be less expressive, as the parallel models may not capture all the nuanced temporal variations in natural speech.

Phoneme duration predictors can be trained using alignments derived from forced alignment methods (Povey et al., 2011), which use dynamic programming to find the optimal alignment between phonemes and corresponding ground truth audio.

The input phonemes are usually processed by an encoder network, which outputs a higher-level representation of the phonemes. These encodings are then upsampled to frame-level representations, typically using a duration predictor, and fed into a decoder network to generate the final Mel-spectrogram values.

The choice of modern encoder architecture often falls to Transformer models (Vaswani

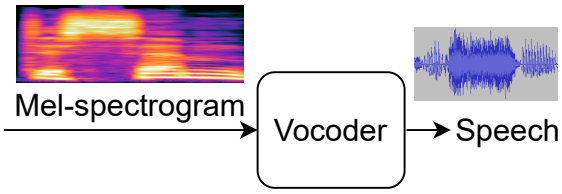


Figure 3.5: Overview of a vocoder module.

et al., 2017), as seen in Ren et al. (2019, 2020), due to their capacity to capture long-range dependencies between phonemes. On the other hand, there is greater variety in decoder strategies. Deterministic convolutional neural networks (CNNs) and Transformer models offer a stable and efficient approach to decoding, while generative models like normalizing flows (Rezende & Mohamed, 2015; Kim et al., 2020) or diffusion models (Popov et al., 2021) can provide greater expressivity by modelling the data distribution, which allows for sampling varied model outputs. For instance, GlowTTS (Kim et al., 2020) contains a normalizing flow module (Prenger et al., 2019) that learns an invertible transformation between Mel-spectrograms and their embeddings, which are regularized to be close in space to the text embeddings from a transformer encoder. The flow learns to embed the Mel-spectrograms during training but it can also perform the inverse operation of converting the text embeddings into Mel-spectrograms during inference.

3.2.6 Vocoders

Vocoders play a critical role in the process of speech synthesis by converting the generated Mel-spectrograms into an audio waveform. However, this task is far from straightforward due to the lossy nature of Mel-spectrograms. The process of converting a spectrogram to a Mel-spectrogram involves downsampling the frequencies to a lower granularity Mel scale. Additionally, phase information is often discarded in spectrograms and acoustic models predict only spectrogram magnitudes, making it necessary for the vocoder to reconstruct the phase aspect of the speech signal.

Griffin-Lim (Griffin & Lim, 1984) is a popular algorithm used to reconstruct phase information from magnitude-only spectrograms. The algorithm iteratively estimates the phase by exploiting redundancies between neighboring spectrogram frames that arise from the overlapping computations in the short-time Fourier transform. The resulting complex-valued spectrograms, which consist of the desired magnitude and estimated phase, can be transformed back into a waveform via an inverse short-time Fourier transform (iSTFT). However, due to the approximations inherent in the Griffin-Lim algorithm, the quality of the reconstructed speech signal often falls short of natural human speech.

As a result, recent advancements in vocoding technology have largely been driven by deep learning models, leading to a range of state-of-the-art vocoders such as auto-regressive models (Oord et al., 2016), normalizing flow models (Oord et al., 2018; Prenger et al., 2019), and diffusion-based models (Chen et al., 2020; Kong et al., 2020b). Nonetheless, as of the

time of writing, the most widely-used vocoders are based on Generative Adversarial Networks (GANs) (Goodfellow et al., 2020), including models such as MelGAN (Kumar et al., 2019), HiFi-GAN (Kong et al., 2020a), and BigVGAN (Lee et al., 2023). These models are favored due to their relatively small size, fast inference speed, and high-quality speech reconstruction.

GAN-based vocoders are characterized by their two-part structure consisting of a generator and a discriminator. The generator is tasked with creating waveforms from Mel-spectrograms that are as indistinguishable as possible from real waveforms. Concurrently, the discriminator is trained to differentiate between these synthesized waveforms and actual waveforms. The training process of these models involves a delicate balance: while the generator strives to fool the discriminator with its generated waveforms, the discriminator continuously improves at identifying the generator's outputs.

Additionally, GAN training stability is typically enhanced with the aid of a feature matching loss. This loss function minimizes the L1-norm difference between the features of the discriminator's output for the generated and real waveforms, helping to guide the generator towards producing waveforms that not only deceive the discriminator but also closely resemble the true speech signals. The interplay between the generator and the discriminator during training leads to a robust and efficient vocoder capable of generating high-quality, natural-sounding speech.

3.2.7 Controllability

The previous sections have largely discussed speech synthesis systems designed for a single speaker, speaking a single language and accent, with neutral prosody/style, and typically recorded in a clean studio environment. However, practical, real-world applications often require systems with much higher controllability, including control over the speaker's identity (timbre), language and accent, prosody/style of the speech, and the sound of the background environment.

Multi-speaker and multi-lingual models provide a more cost-effective solution for hosting large-scale speech synthesis services as they eliminate the need to maintain individual models for each speaker or language. Moreover, these models offer the potential for novel applications, such as synthesizing speech in a particular speaker's voice but in a language not seen during training for that speaker (Jia et al., 2018). However, achieving this requires successful disentanglement of speaker identity and language information to prevent unwanted accents in the synthesized speech.

Locale embeddings, which encode language and accent information, are typically modelled as one-hot vectors in these systems. Similarly, speaker identity can be encoded either as a one-hot vector or extracted from a reference audio clip of the target speaker (Wu et al., 2022). The latter approach offers the possibility of synthesizing speech for unseen speakers, but its effectiveness depends on the system's ability to disentangle the speaker's voice characteristics from other aspects of the reference audio.

In many applications, it is desirable to synthesize speech with a specific style or emotional tone. For example, a virtual assistant might be required to speak empathetically, or a dubbed audio track might need to match the emotional tone of the original performance. This can

be achieved either by using discrete emotion labels (Rattcliffe et al., 2022a), or by extracting continuous emotion/style embeddings from a reference audio (Skerry-Ryan et al., 2018b; Wang et al., 2018). The latter approach offers the advantage of capturing a broader range of emotions and does not require emotion labels, but its success again hinges on the system's ability to disentangle the prosodic properties of the reference audio.

Recording studio-quality speech is a costly and time-consuming process, which can limit the amount of data available for training. With the abundance of readily accessible, albeit noisy, data on the internet, it is advantageous to train robust speech synthesis systems using such data while still being able to produce clean, studio-quality speech when necessary. This necessitates the disentanglement of background noise from the speech signal itself (Zhang et al., 2021).

A common way to achieve the above-described controllability is through learning disentangled representations of speech that can be conditioned on the control factors as we described in Chapter 6. However, most recent works have explored also an alternative approach. These methods achieve control over voice identity, language, prosody, and background effects through prompting, either by auto-regressive continuation (Wang et al., 2023) or in-painting with a diffusion model (Shen et al., 2023) or a transformer model (Borsos et al., 2023b).

In conclusion, the level of controllability over various aspects of the synthesized speech is a key requirement for practical applications, and recent research has made significant strides in addressing this challenge through a variety of strategies, whether through disentanglement or innovative prompting techniques. In the next section, we will describe Variational Auto-Encoders (VAEs) and their use in speech synthesis such as improving prosody controllability and disentanglement.

3.3 Variational Auto-Encoder (VAE)

The Variational Auto-Encoder (VAE) is a cornerstone of variational inference applied to deep learning, serving as a potent tool for unsupervised learning tasks, particularly in the representation learning of complex data distributions. Introduced by Kingma & Welling (2013), VAEs are a class of generative models that are particularly effective for tasks where we aim to capture, understand and generate high-dimensional data (Kingma & Welling, 2013; Rezende et al., 2014).

VAEs are based on autoencoders (Hinton & Salakhutdinov, 2006; Goodfellow et al., 2016), a type of neural network architecture designed for unsupervised learning of data encodings. An autoencoder consists of two parts: an encoder, which maps input data to a lower-dimensional representation (also known as latent space), and a decoder, which attempts to reconstruct the original input from the lower-dimensional representation. While the vanilla autoencoder is trained to minimize the reconstruction error, the VAE differs in its incorporation of probabilistic modelling, utilizing variational inference.

In a VAE, the encoder is not just mapping input data to a fixed point in the latent space; instead, it's learning the parameters of a probability distribution representing the data. More precisely, given an input data point $\mathbf{x}$, the encoder of the VAE models the parameters of the

posterior distribution $p(\mathbf{z}|\mathbf{x})$, where $\mathbf{z}$ is the latent variable.

The decoder, on the other hand, is designed to reconstruct the original input $\mathbf{x}$ from samples drawn from the encoded distribution $p(\mathbf{z}|\mathbf{x})$. It does so by modelling the likelihood distribution $p(\mathbf{x}|\mathbf{z})$. This results in a more continuous latent space compared to the standard autoencoders, which improves latent space interpolation and generalizability (Kingma & Welling, 2013; Bowman et al., 2015). As discussed in Section 1.5, the posterior $p(\mathbf{z}|\mathbf{x})$ for the complex models is intractable due to the computation of the posterior normalizing constant $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, d\mathbf{z}$. In the case of VAEs, this would involve integrating over all possible latent space values $\mathbf{z}$. Therefore, VAEs employ the variational approximation $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ of the true posterior $p(\mathbf{z}|\mathbf{x})$.

In VAEs the variational posterior $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is over the latent representation $\mathbf{z}$ of input $\mathbf{x}$. Additionally, the optimized variational parameters are the weights of a neural network that predicts the variational distribution parameterization. In comparison, in Bayesian neural networks (BNNs), the variational posterior $q_{\boldsymbol{\theta}}(\mathbf{w})$ is over neural network weights $\mathbf{w}$, and the optimized variational parameters are directory the posterior distribution parameterizations (e.g. $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in GMFVI BNNs from Section 2.2.1). Therefore, VAE posterior usually models a smaller number of dimensions compared to BNNs ($\dim(\mathbf{z}) \ll \dim(\mathbf{w})$), but the VAE posterior has more capacity in modelling complex distributions. Furthermore, note that the VAE variational posterior $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is conditional on both variational parameters $\boldsymbol{\theta}$ and the neural network input $\mathbf{x}$. In Bayesian neural networks, the variational posterior $q_{\boldsymbol{\theta}}(\mathbf{w})$ is conditional only on the training data through the variational parameters $\boldsymbol{\theta}$ trained using amortized inference (Gershman & Goodman, 2014).

3.3.1 Gaussian Mean-Field Variational Auto-Encoder

A common choice for the variational distribution q_θ in VAEs is the multivariate Gaussian with diagonal covariance, termed as Gaussian Mean-Field Variational Inference (GMFVI) previously in the context of Bayesian neural networks from Section 1.5.1. GMFVI factorises dimensions of the latent space $\mathbf{z}$ by making an independence assumption. In this setting, the encoder network predicts the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ parameters of $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ for a given input $\mathbf{x}$. Latent vector $\mathbf{z}$ is then sampled using the reparameterization trick $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$. The reparametrization allows to backpropagate gradients through the sampling operation, making it feasible to train the VAE using gradient-based optimization techniques. Finally, the latent $\mathbf{z}$ is passed to the decoder $p_\phi(\mathbf{x}|\mathbf{z})$. Figure 3.6 illustrates this approach.

The VAE is optimized using the evidence lower bound (ELBO) objective introduced in Section 1.5. For VAEs, the negative ELBO takes the form:

$$Neg.ELBO = D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] - \mathbb{E}_q\left[\log p_\phi(\mathbf{x}|\mathbf{z})\right] \tag{3.1}$$

Equation 3.1 reflects the balance between data fidelity (reconstruction loss $\mathbb{E}_q\left[\log p_\phi(\mathbf{x}|\mathbf{z})\right]$) and the complexity of the model (regularization term $D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]$). The regularization term is essentially the Kullback-Leibler (KL) divergence between the approximate posterior

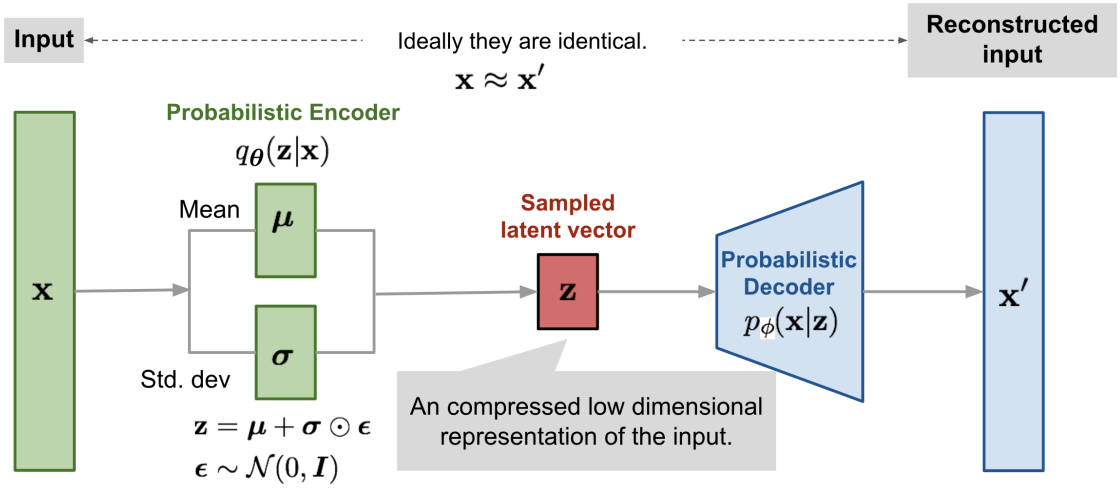


Figure 3.6: Illustration of variational autoencoder model with the diagonal multivariate Gaussian assumption. Source: Weng (2018).

distribution and the prior distribution, which in most applications, is assumed to be a standard normal distribution. This term encourages the network to use the latent space efficiently, resulting in learned representations that are often more interpretable, disentangled and structured (Bowman et al., 2015; Zhang et al., 2019c).

3.3.2 Practical challenges with training VAEs

The practical challenges inherent in training Variational Auto-Encoders (VAEs) should not be overlooked. A specific challenge to note is the well-documented tendency of the posteriors within VAEs to collapse towards a prior distribution (Sønderby et al., 2016). This typically occurs due to the confluence of the powerful, uniform gradient originating from the regularization term of the Evidence Lower Bound (ELBO) objective and the noisy gradients of the data fit term. Consequently, the model is often entrapped in local minima where the inputs, denoted as $\mathbf{x}$, are disregarded, resulting in the encoder network invariably predicting a posterior distribution identical to the prior.

This situation draws parallels to the *cold posterior* effect, a phenomenon introduced in Section 2.5 and thoroughly examined in Chapter 5 within the framework of Bayesian Neural Networks (BNNs). Both in the context of BNNs and VAEs, the mitigation of the regularization Kullback-Leibler (KL) term—equating to a reduction in the temperature of the posterior distribution — can lead to enhanced model performance. An alternative approach to this issue involves the application of an annealing strategy to the KL weight contribution (Sønderby et al., 2016). By gradually escalating the contribution from 0 to 1 at the onset of training, the likelihood term is afforded an opportunity to converge within its local minima prior to the introduction of the KL divergence regularization, thereby providing

a countermeasure to the posterior collapse.

In conclusion, despite the practical challenges, VAEs provide a powerful and versatile model for unsupervised learning, capable of learning rich, latent representations of data. Their theoretical foundation in variational inference makes them particularly suited for tasks where we want to both generate new samples from learned data distribution and understand the underlying structure of our data. In the following sections, we'll explore how these properties of VAEs can be leveraged in the field of speech synthesis.

3.4 VAE in Speech Synthesis

Utilization of Variational Auto-Encoders (VAEs) extends into several realms of speech synthesis, chiefly because of their capacity to learn continuous, generalizable, and disentangled representations. Within this context, we elucidate two salient applications of VAEs. Firstly, we delve into the exploration of VAEs as proficient methods for acquiring prosody representations with transferable attributes across multiple utterances. Secondly, we discuss the capability of VAEs to learn more comprehensive audio representations, offering an advantageous alternative to Mel-spectrograms as an intermediary modeling representation within the architectural framework of speech synthesis delineated in the preceding section.

3.4.1 VAE for Prosody Transfer

Continuous prosody encoders for reference audio, as mentioned in Section 3.2.7, offer considerable advantages such as the unsupervised learning of representations and the elimination of the requirement for prosody labels. An initial implementation of this approach, as proposed in Skerry-Ryan et al. (2018b), employed a Convolutional Neural Network (CNN) down-sampling encoder which coupled with an LSTM to distill the prosody into a single global embedding for each utterance. This embedding was subsequently utilized as a conditioning factor within an acoustic model.

In an effort to enhance this approach, Zhang et al. (2019c) proposed the application of a variational inference framework to the reference encoder. This approach gave rise to a more disentangled and continuous space emanating from the Variational Auto-Encoder (VAE), as described in Bowman et al. (2015), thereby facilitating superior transferability of embeddings across distinct utterances. Zhang et al. (2019c) further demonstrated that these resulting embeddings could be transferred between different speakers, effectively evidencing the successful disentanglement of prosody from speaker information in the embeddings.

Related to the discussion in Section 3.3.2, it is crucial to precisely calibrate the Kullback-Leibler (KL) term in the VAE's Evidence Lower Bound (ELBO) objective weights in order to achieve optimal transfer and disentanglement. A disproportionately high contribution from the KL term can induce the posterior to collapse to a prior distribution, resulting in the model ignoring the reference audio. Conversely, insufficient KL regularization can lead to entanglement of the prosody with content and speaker information in the embeddings derived from the reference audio. An appropriately tuned scaling of the VAE's KL term

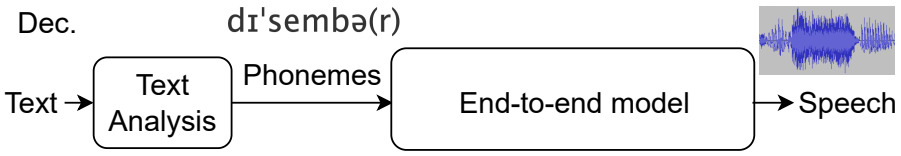


Figure 3.7: Overview of an end-to-end TTS system.

encourages the synthesis model to exclusively extract prosody information from the reference audio, while acquiring content information from the input phoneme sequence and speaker information from the provided speaker embeddings.

3.4.2 VAE for Audio Representation Learning

Until recently, the two-stage architectures with separate acoustic and vocoder models described in Section 3.2.3, were the leading performers in the field of speech synthesis. However, this two-stage architecture is not without its drawbacks. The primary issue lies in the data shift between the Mel-spectrograms forecasted by the acoustic model, which are often overly smoothed, and the ground truth Mel-spectrograms used for training vocoders. This discrepancy necessitates further fine-tuning of the already trained vocoders on the predicted Mel-spectrograms to attain optimal performance. Furthermore, the Mel-spectrogram does not serve as the ideal intermediate representation as it is a hand-crafted feature and it is not learned.

VITS (Kim et al., 2021) is the first end-to-end model combining the acoustic and vocoder modules into a single model, which out-performed the two-stage architectures (Figure 3.7). VITS accomplishes this by employing a Variational Auto-Encoder (VAE) to learn an internal audio representation. The encoder of VITS's VAE is based on WaveNet (Oord et al., 2016) blocks, and the decoder of VITS is a HiFi-GAN, as introduced in section 3.2.6. The VITS's VAE is concurrently trained with VITS's acoustic model, based on GlowTTS (Kim et al., 2020), which we introduced in Section 3.2.5. In VITS, the GlowTTS flow predicts the VAE representation from the linguistic content, instead of predicting the Mel-spectrogram as proposed originally. Figure 3.8 shows an overview of a modified VITS architecture with the combined VAE and acoustic model that we adopt in Chapters 6 and 7.

VITS demonstrated exceptional results, achieving human-equivalent naturalness in the synthesized speech. However, a significant drawback of VITS is the extended training time required for each iteration of the model. The training of the HiFi-GAN vocoder component, in particular, is time-consuming due to the extensive sequence lengths of raw audio and the objectives of GAN training. This protracted training process hinders rapid experimentation.

More recently, a trend prevalent in the generative computer vision community where models operate on VAE learned latent representations (Rombach et al., 2022), has begun to permeate the audio generation domain. Notably, state-of-the-art speech synthesis methods are now utilizing Residual Vector Quantized Variational Auto-encoders with Adversarial Learning (Zeghidour et al., 2021; Défossez et al., 2022). These audio codecs yield high-quality

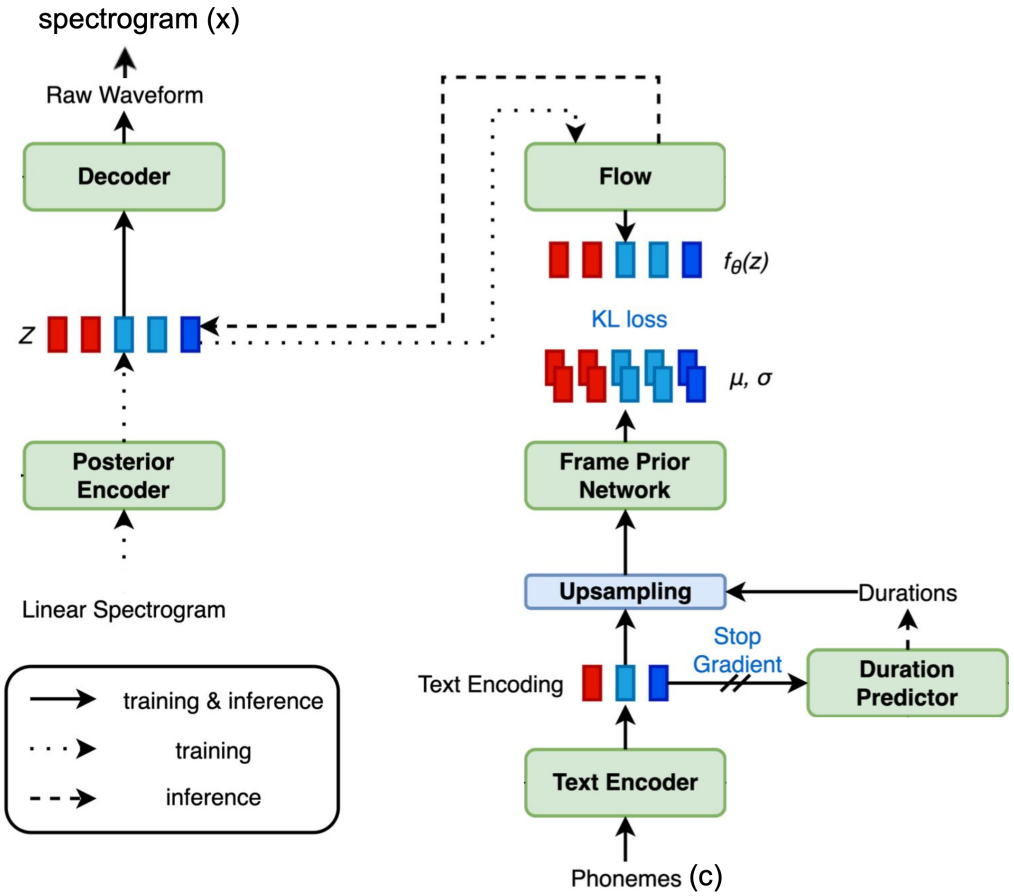


Figure 3.8: Overview of a modified VITS architecture that we adopt in Chapters 6 and 7. Two modifications visible here compared to the original VITS from Kim et al. (2021) are the use of a duration predictor as described in Section 3.2.5 and an additional frame prior network that smooths upsampled text encodings, both as proposed in Zhang et al. (2022).

audio reconstruction and can serve as targets for other models. Due to their discrete nature, they can be partnered with transformer models (Vaswani et al., 2017) for modeling the audio codes (Borsos et al., 2023a; Wang et al., 2023; Borsos et al., 2023b).

In the next section, we describe the task of machine dubbing for which the above applications of VAE are employed in a practical setting, as later summarized in Sections 3.6 and 3.7, and described in detail in Chapters 6 and 7.

3.5 Machine Dubbing

Machine dubbing involves synthesizing translated speech from a given reference video containing original language speech. The task necessitates the translated speech to reflect the

vocal performance observed in the original language. Achieving satisfactory performance in machine dubbing involves overcoming several challenges and fulfilling multiple requirements.

Firstly, despite the unprecedented naturalness exhibited by state-of-the-art synthesis systems, they struggle to generate the expressive and spontaneous speech characteristic of multimedia recordings.

Secondly, the system must maintain precise control over expressiveness to ensure it aligns with the original vocal performance.

Thirdly, the duration of the synthesized speech must correlate with the lip movements visible in the underlying video, ensuring synchronicity between auditory and visual elements.

Lastly, multimedia recordings are frequently captured in uncontrolled environments, which often introduce background noise. This stands in contrast to conventional text-to-speech datasets which are recorded in controlled, clean studio environments. Such noisy data necessitate learning methods capable of handling the inherent noise and producing clean dubbed speech. As such, the system must be equipped with mechanisms to disentangle prosody from noise in the reference recordings (Zhang et al., 2021).

The Variational Auto-Encoder (VAE) framework, outlined in Section 3.4, offers an effective solution to address these challenges. The remaining two sections in this chapter will summarize two published works that propose machine dubbing systems grounded in VAEs, demonstrating their capability to satisfy the aforementioned requirements.

3.6 Summary of Chapter 6

In this section, we provide a concise summary of the contributions and results of paper P3, which is later described in more detail in Chapter 6.

Prosody transfer within a single language has been extensively studied in the context of expressive speech synthesis (Skerry-Ryan et al., 2018b; Wang et al., 2018; Zhang et al., 2019c). However, the field of cross-lingual prosody transfer has received less attention to date.

Notably, previous machine dubbing systems did not effectively transfer prosody from original speech (Federico et al., 2020; Virkar et al., 2021). Therefore, in Chapter 6, we introduce a machine dubbing system capable of transferring prosody across both languages and speakers.

Our method, termed Variational Inference for Prosody Transfer (VIPT), augments the VITS system (Kim et al., 2021) detailed in Section 3.4.2 with a VAE-based prosody encoder, explained in section 3.4.1, and additional language and speaker conditioning, as shown in Figure 3.9. Consequently, our system learns to disentangle prosody representations that can be effectively transferred across languages and speakers.

Furthermore, we devise a method for disentangling noise that enables our system to synthesize clean dubs even when the reference audio is noisy. Our method, depicted in Figure 3.10, separates denoised and noise streams via a pre-trained denoiser component (Isik et al., 2020), which are then processed by distinct VAE reference encoders that condition the

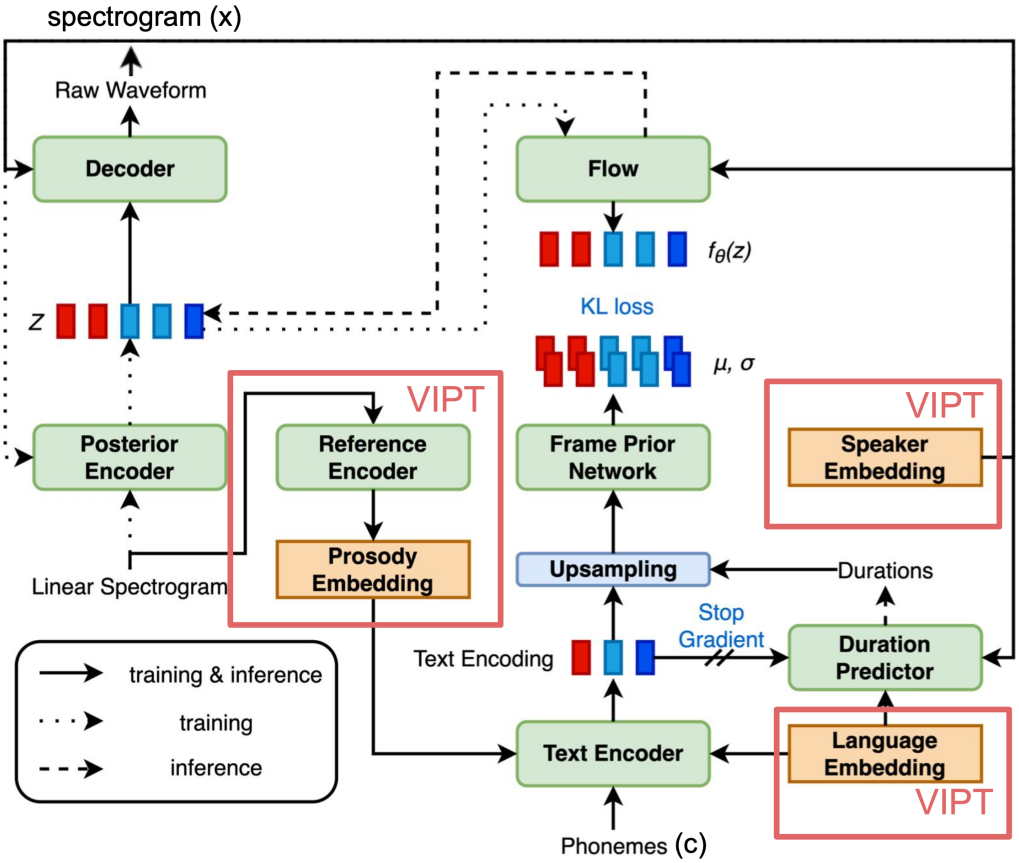


Figure 3.9: Proposed VIPT architecture. Our VIPT extensions to VITS are marked in red squares.

VITS's acoustic model. During inference, we substitute the noise stream with clean audio, which results in clean speech synthesis.

Our refined system bridges the gap between a baseline that lacks cross-lingual prosody transfer and human recordings by 11%, as illustrated in Figure 3.11.

3.7 Summary of Chapter 7

In this section, we provide a concise summary of the contributions and results of paper P4, which is later described in more detail in Chapter 7.

In Chapter 7, we present a method for phrase-level cross-lingual prosody transfer, enhancing the expressivity of multi-lingual machine dubbing compared to the VIPT method summarised in Section 3.6. The improved approach models and transfers VAE-based prosody representations at the level of prosodic phrases, which are defined as continuous speech segments separated by silence regions.

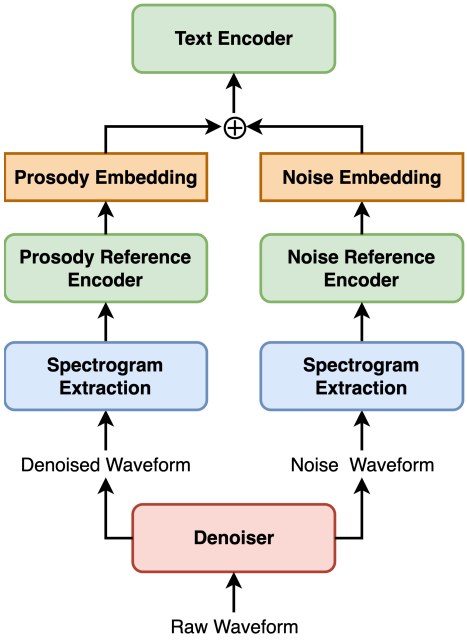


Figure 3.10: Illustration of our noise modelling method in VIPT that conditions text encoder module.

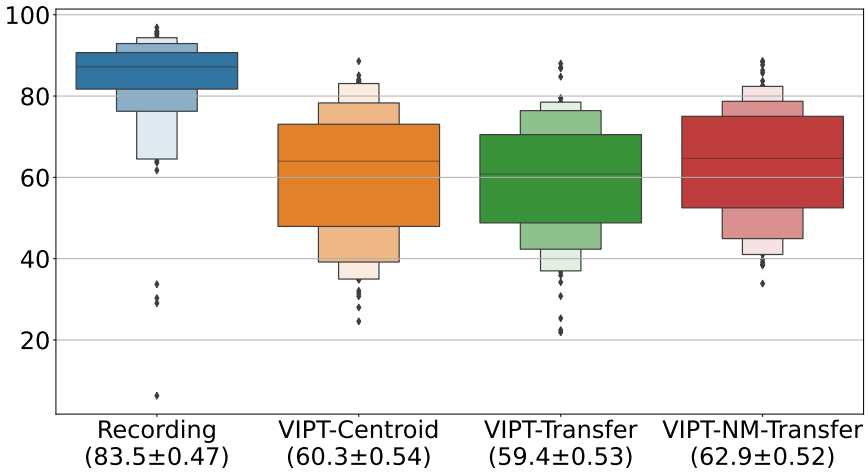


Figure 3.11: Subjective listener ratings from a machine dubbing MUSHRA test (Recommendation, 2015) on an expressive and noisy multimedia dataset. While cross-lingual prosody transfer using VIPT method without noise modelling (VIPT-Transfer) performs worse compared to a baseline VIPT without the cross-lingual prosody transfer (VIPT-Centroid), cross-lingual prosody transfer with VIPT containing a noise modelling module (VIPT-NM-Transfer) beats the baseline.

VIPT introduced a method for global prosody transfer at the utterance level. However, it was not equipped to encode detailed local prosody variations, which are crucial for expressive machine dubbing (Torresquintero et al., 2021).

Chapter 7 addresses this limitation with a phrase-level prosody transfer, which modifies the VAE-based prosody reference encoder employed in VIPT. This phrase-level reference encoder is designed to condition the phrases of the input text with prosody embeddings extracted from corresponding segments of the reference speech waveform, as illustrated in Figure 3.12.

The more granular modelling of prosody allows a greater capacity for prosody representation. However, it also poses more significant challenges in disentangling prosody from other factors, such as content information. This issue is particularly evident in shorter prosodic phrases, where a single prosody embedding can easily encode the content information. To address this, Chapter 7 proposes a length-based regularization of the phrase-level VAE reference encoder. This approach results in stronger regularization for shorter phrases, encouraging the model to rely more on the content information from the text input.

The proposed phrase-level prosody transfer delivers a significant 6.2% increase in the subjective listening scores over a baseline with utterance-level global prosody transfer. Thus, it bridges the gap between the baseline and expressive human dubbing by 23.3%, as illustrated in Figure 3.13. This improvement is confirmed by the objective conditional Frechet DeepSpeech Distance (cFDSD) metric (Bińkowski et al., 2020), as shown in Table 3.1.

Lastly, P4 demonstrates the effectiveness of the proposed length-based regularization for short phrases, as displayed in Table 3.1.

Table 3.1: *Objective metrics: word error rate (WER) and conditional Fréchet DeepSpeech Distance (cFDSD) Bińkowski et al. (2020).*

System	**cFDSD ↓**	**WER ↓**	
		all	shortest 25%
VIPT-NM-GVAE	0.297	0.098	0.169
VIPT-NM-GVAE-PP	0.288	0.094	0.155
VIPT-NM-PVAE	**0.224**	0.101	0.161
w/o length-based reg.	0.241	0.106	0.229

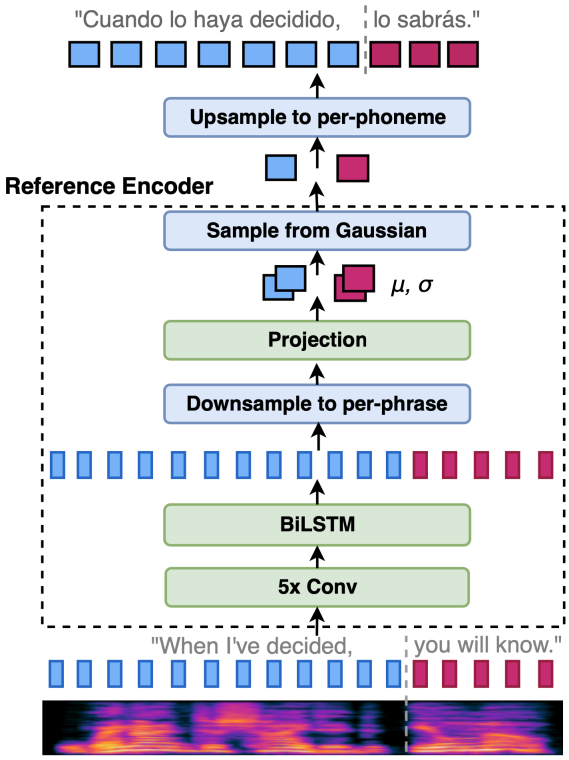


Figure 3.12: Illustration of a phrase-level reference encoder in the context of cross-lingual prosody transfer as in machine dubbing.

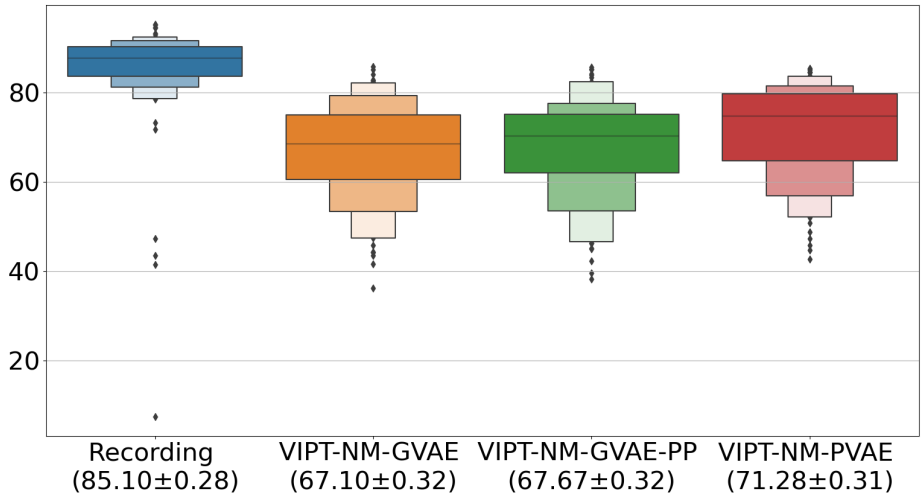


Figure 3.13: Subjective listener ratings from a machine dubbing MUSHRA test (Recommendation, 2015) on a expressive and noisy multimedia dataset. Our proposed system trained with a phrase-level reference encoder (VIPT-NM-PVAE) is rated better than systems trained with a global utterance-level reference encoder (VIPT-NM-GVAE and VIPT-NM-GVAE-PP). VIPT-NM-GVAE-PP performs phrase-level prosody transfer at inference time while being trained as a global-level reference encoder.

Chapter 4

The k-tied Normal Distribution: A Compact Parameterization of Gaussian Mean Field Posteriors in Bayesian Neural Networks

4.1 Overview

Variational Bayesian Inference is a popular methodology for approximating posterior distributions over Bayesian neural network weights. Recent work developing this class of methods has explored ever richer parameterizations of the approximate posterior in the hope of improving performance. In contrast, here we share a curious experimental finding that suggests instead restricting the variational distribution to a more compact parameterization. For a variety of deep Bayesian neural networks trained using Gaussian mean-field variational inference, we find that the posterior standard deviations consistently exhibit strong low-rank structure after convergence. This means that by decomposing these variational parameters into a low-rank factorization, we can make our variational approximation more compact without decreasing the models' performance. Furthermore, we find that such factorized parameterizations improve the signal-to-noise ratio of stochastic gradient estimates of the variational lower bound, resulting in faster convergence.

4.2 Introduction

We introduced Gaussian Mean-Field Variational Inference (GMFVI) for Bayesian neural networks (BNNs) in Section 2.2 of this thesis. Beyond mean-field variational inference, recent work on approximate Bayesian inference has explored ever richer parameterizations of the approximate posterior in the hope of improving the performance of Bayesian neural networks (see Figure 4.1). In contrast, here we study a simpler, more compactly parameterized

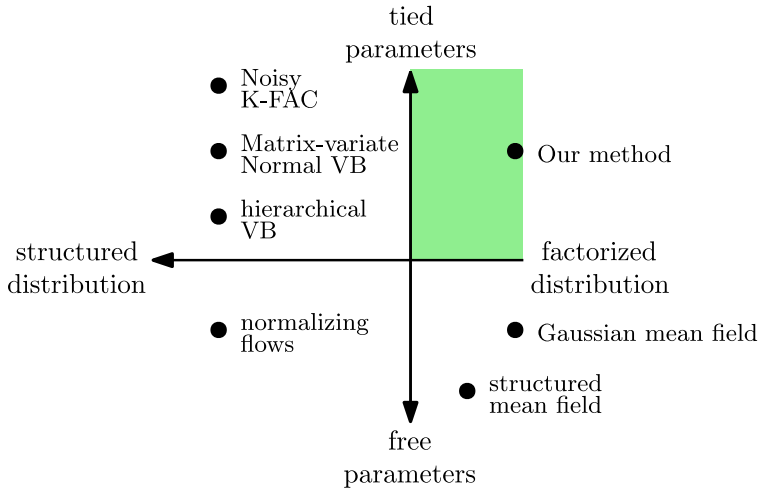


Figure 4.1: Approximate summarization of different variational inference methods for Bayesian deep learning. Our approach complements existing approaches by combining the mean-field assumption with a dramatic reduction in the number of parameters by weight sharing.

variational approximation. Our motivation for studying this setting is to better understand the behaviour of GMFVI with the goal to address the issues with its practical applicability. Consequently, we show that the compact approximations can also work well for a variety of models. In particular we find that:

- Converged posterior standard deviations under GMFVI consistently display strong low-rank structure. This means that by decomposing these variational parameters into a low-rank factorization, we can make our variational approximation more compact without decreasing our model's performance.
- Factorized parameterizations of posterior standard deviations improve the signal-to-noise ratio of stochastic gradient estimates, and thus not only reduce the number of parameters compared to standard GMFVI, but also can lead to faster convergence.

4.3 Mean Field Posterior Standard Deviations Naturally Have Low-Rank Structure

In this section, we show that the converged posterior standard deviations of Bayesian neural networks trained using standard GMFVI consistently display strong low-rank structure. We also show that it is possible to compress the learned posterior standard deviation matrix using a low-rank approximation without decreasing the network's performance. We first briefly reintroduce the mathematical notation for our GMFVI setting from Section 1.5.1 and the low-rank approximation as summarized in Section 2.4. We then provide experimental results that support the two main claims of this section.

To avoid any confusion among the readers, we would like to clarify that we use the

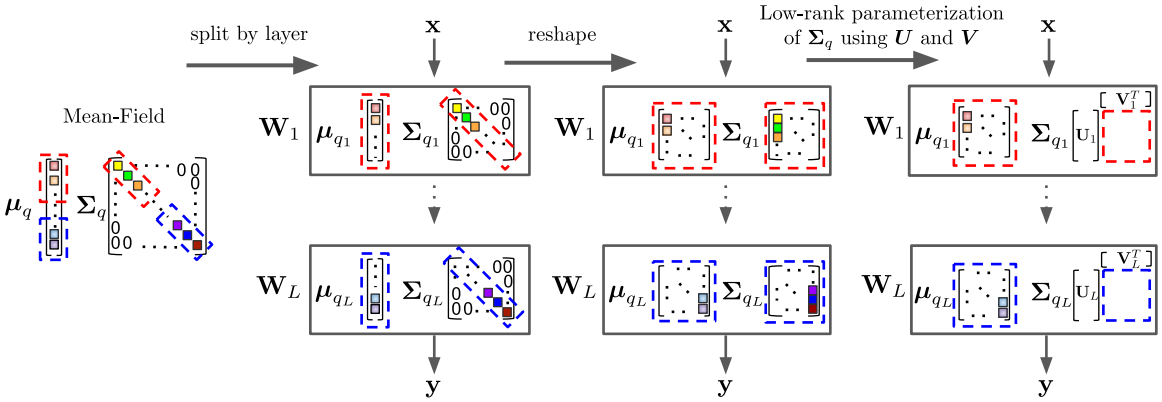


Figure 4.2: Illustration of the relationship between the standard Gaussian Mean-Field posterior and its "low-rank" parameterization, which we call the k-tied Normal posterior. The illustration shows the posterior parameterization for a network with L layers, where $\mathbf{x}$ and $\mathbf{y}$ are the network inputs and outputs respectively, and $\boldsymbol{\mu}_{q_1}$, $\boldsymbol{\Sigma}_{q_1}$, $\boldsymbol{\mu}_{q_L}$ and $\boldsymbol{\Sigma}_{q_L}$ are the variational parameters for the layers 1 and L respectively. The k-tied Normal distribution parameterizes the already diagonal per layer posterior covariance matrices $\boldsymbol{\Sigma}_{q_{1..L}}$ using the even more compact $\boldsymbol{U}_{1..L}$ and $\boldsymbol{V}_{1..L}$ matrices from $\mathcal{N}\big(\boldsymbol{\mu}_q, \mathrm{diag}\big(\mathrm{vec}\big((\mathbf{U}\mathbf{V}^T)^2\big)\big)\big)$.

terminology "low-rank" in a particular context. While variational inference typically makes use of low-rank decompositions to compactly represent the *dense covariance* of a Gaussian variational distribution (see numerous references in Section 4.5), we investigate instead underlying low-rank structures within *the already diagonal covariance* of a Gaussian fully-factorized variational distribution. Figure 4.2 aims to make this even more clear by illustrating the relationship between the Gaussian fully-factorized variational distribution and its "low-rank" parameterization explored in this chapter. We will make this explanation more formal in the next section.

4.3.1 Methodology

To introduce the notation, we consider layers that consist of a linear transformation followed by a non-linearity f,

$$\mathbf{a}_l = \mathbf{h}_l \mathbf{W}_l + \mathbf{b}_l, \qquad \mathbf{h}_{l+1} = f(\mathbf{a}_l), \tag{4.1}$$

where $\mathbf{W}_l \in \mathbb{R}^{m \times n}$, $\mathbf{h}_l \in \mathbb{R}^{1 \times m}$ and $\mathbf{b}_l, \mathbf{a}_l, \mathbf{h}_{l+1} \in \mathbb{R}^{1 \times n}$. To simplify the notation in the following, we drop the subscript l such that $\mathbf{W} = \mathbf{W}_l$, $\boldsymbol{\mu}_q = \boldsymbol{\mu}_{ql}$, $\boldsymbol{\Sigma}_q = \boldsymbol{\Sigma}_{ql}$ and we focus on the kernel matrix $\mathbf{W}$ for a single layer.

In GMFVI, we model the variational posterior as

$$\begin{aligned} q(\mathbf{W}) = \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) = \prod_{i=1}^{m} \prod_{j=1}^{n} q(w_{ij}), \\ \text{with} \quad q(w_{ij}) = \mathcal{N}(\mu_{ij}, \sigma_{ij}^2), \end{aligned} \tag{4.2}$$

Variational family	Parameters (total)
multivariate Normal	$mn + \frac{mn\,(mn+1)}{2}$
diagonal Normal	$mn + mn$
$\mathcal{MN}$	$mn + \frac{m(m+1)}{2} + \frac{n(n+1)}{2}$
$\mathcal{MN}$(diagonal)	$mn + m + n$
k-tied Normal (ours)	$mn + k(m+n)$

Table 4.1: Number of variational parameters for a variational family for a matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$. $\mathcal{MN}$(diagonal) is from Louizos & Welling (2016).

where $\boldsymbol{\mu}_q \in \mathbb{R}^{mn\times 1}$ is the posterior mean vector, $\boldsymbol{\Sigma}_q \in \mathbb{R}_+^{mn\times mn}$ is the diagonal posterior covariance matrix. The weights are then usually sampled using a reparameterization trick (Kingma & Welling, 2013), i.e, for the s-th sample, we have

$$w_{ij}^{(s)} = \mu_{ij} + \sigma_{ij}\epsilon^{(s)}, \qquad \epsilon \sim \mathcal{N}(0,1). \tag{4.3}$$

In practice, we often represent the posterior standard deviation parameters σ_{ij} in the form of a matrix $\mathbf{A} \in \mathbb{R}_+^{m\times n}$. Note that we have the relationship $\boldsymbol{\Sigma}_q = \mathrm{diag}(\mathrm{vec}(\mathbf{A}^2))$ where the elementwise-squared $\mathbf{A}$ is vectorized by stacking its columns, and then expanded as a diagonal matrix into $\mathbb{R}_+^{mn\times mn}$.

In the sequel, we start by empirically studying the properties of the spectrum of matrices $\mathbf{A}$ *post-training* (after convergence), while using standard Gaussian mean-field variational distributions. Interestingly, we observe that those matrices naturally exhibit a low-rank structure (see Section 4.3.3 for the corresponding experiments), i.e,

$$\mathbf{A} \approx \mathbf{U}\mathbf{V}^T \tag{4.4}$$

for some $\mathbf{U} \in \mathbb{R}^{m\times k}$, $\mathbf{V} \in \mathbb{R}^{n\times k}$ and k a small value (e.g., 2 or 3). This observation motivates the introduction of the following variational family, which we name k-tied Normal:

$$\boxed{\begin{array}{c} k\text{-tied-}\mathcal{N}(\mathbf{W};\boldsymbol{\mu}_q,\mathbf{U},\mathbf{V}) = \\ \mathcal{N}\big(\boldsymbol{\mu}_q, \mathrm{diag}\big(\mathrm{vec}\big((\mathbf{U}\mathbf{V}^T)^2\big)\big)\big), \end{array}} \tag{4.5}$$

where the squaring of the matrix $\mathbf{U}\mathbf{V}^T$ is applied elementwise. Due to the tied parameterization of the diagonal covariance matrix, we emphasize that this variational family is *smaller*—i.e., included in—the standard Gaussian mean-field variational distribution family.

As formally discussed in Appendix A.1, the matrix variate Gaussian distribution (Gupta & Nagar, 2018), referred to as $\mathcal{MN}$ and already used for variational inference by Louizos & Welling (2016) and Sun et al. (2017), is related to our k-tied Normal distribution with $k = 1$ when $\mathcal{MN}$ uses diagonal row and column covariances. Interestingly, we prove that for $k \geq 2$, our k-tied Normal distribution cannot be represented by any $\mathcal{MN}$ distribution. This illustrates the main difference of our approach from the most closely related previous work of Louizos & Welling (2016).

Notice that our diagonal covariance $\mathbf{\Sigma}_q$ repeatedly reuses the same elements of $\mathbf{U}$ and $\mathbf{V}$, which results in parameter sharing across different weights. The total number of the standard deviation parameters in our method is $k(m+n)$ from $\mathbf{U}$ and $\mathbf{V}$, compared to mn from $\mathbf{A}$ in the standard GMFVI parameterization. Given that in our experiments the k is very low (e.g. $k=2$) this reduces the number of parameters from quadratic to linear in the dimensions of the layer, see Table 4.1. More importantly, such parameter sharing across the weights leads to higher signal-to-noise ratio during training and thus in some cases faster convergence. We demonstrate this phenomena in the next section. In the rest of this section, we show that the standard GMFVI methods already learn a low-rank structure in the posterior standard deviation matrix $\mathbf{A}$. Furthermore, we provide evidence that replacing $\mathbf{A}$ with its low-rank approximation does not degrade the predictive performance and the quality of uncertainty estimates.

4.3.2 Experimental setting

Before describing the experimental results, we briefly explain the key properties of the experimental setting. We analyze three types of GMFVI Bayesian neural network models:

- Multilayer Perceptron (MLP): a network of 3 dense layers and ReLu activations that we train on the MNIST dataset (LeCun & Cortes, 2010). We use the last 10,000 examples of the training set as a validation set.
- Convolutional Neural Network (CNN): a LeNet architecture (LeCun et al., 1998) with 2 convolutional layers and 2 dense layers that we train on the CIFAR-100 dataset (Krizhevsky et al., 2009b). We use the last 10,000 examples of the training set as a validation set.
- Long Short-Term Memory (LSTM): a model that consists of an embedding and an LSTM cell (Hochreiter & Schmidhuber, 1997), followed by a single unit dense layer. We train it on the IMDB dataset (Maas et al., 2011), in which we use the last 5,000 examples of the training set as a validation set.
- Residual Convolutional Neural Network (ResNet): a ResNet-18[1] architecture (He et al., 2016b) trained on all 50,000 training examples of the CIFAR-10 dataset (Krizhevsky et al., 2009a).

In each of the four models, we use the standard mean-field Normal variational posterior and a Normal prior, for which we set a single scalar standard deviation hyper-parameter shared by all layers. Appendix A.2 contains an ablation study result with an alternative prior. We optimize the variational posterior parameters using the Adam optimizer (Kingma & Ba, 2014). For a more comprehensive explanation of the experimental setup please refer to Appendix A.4. Finally, we highlight that our experiments focus primarily on the comparison across a broad range of model types rather than competing with the state-of-the-art results over the specifically used datasets. Nevertheless, we also show that our results extend to larger models with competitive performance such as the ResNet-18 model. Note that scaling

[1]See: `https://github.com/tensorflow/probability/blob/master/tensorflow_probability/examples/cifar10_bnn.py`.

GMFVI to such larger model sizes is still a challenging research problem (Osawa et al., 2019b).

4.3.3 Main experimental observation

Our main experimental observation is that the standard GMFVI learns posterior standard deviation matrices that have a low-rank structure across different model types (MLP, CNN, LSTM), model sizes (LeNet, ResNet-18) and layer types (dense, convolutional). To show this, we investigate the results of the SVD decomposition of posterior standard deviation matrices $\mathbf{A}$ in the four described models types. We analyze the models *post-training*, where the models are already trained until ELBO convergence using the standard GMFVI approach. While for the first three models (MLP, CNN and LSTM), we evaluate the low-rank structure only in the dense layers, for the ResNet model we consider the low-rank structure in the convolutional layers as well.

Figure 4.3 shows the fraction of variance explained per each singular value k from the SVD decomposition of matrices $\mathbf{A}$ in the dense layers of the first three models. The fraction of variance explained per singular value k is calculated as $\gamma_k^2 / \sum_{i'} \gamma_{i'}^2$, where γ are the singular values. We observe that, unlike posterior means, the posterior standard deviations have most of their variance explained by the first few singular values. In particular, a rank-1 approximation of $\mathbf{A}$ explains most of its variance, while a rank-2 approximation can encompass nearly all of the remaining variance. Figure 4.4 further supports this claim visually by comparing the heat maps of the matrix $\mathbf{A}$ and its rank-1 and rank-2 approximations. In particular, we observe that the rank-2 approximation heat map looks visually very similar to $\mathbf{A}$, while this is not the case for the rank-1 approximation. Importantly, Figure 4.5 illustrates that the same low-rank structure is also present in both the dense and the convolutional layers of the larger ResNet-18 model. In the analysis of the above experiments, we use the shorthand SEM to refer to the standard error of the mean.

4.3.4 Low-rank approximation of mean field posterior standard deviations

Motivated by the above observations, we show that it is possible to replace the reshaped diagonal posterior standard deviation matrices $\mathbf{A}$ with their low-rank approximations without decreasing predictive performance and the quality of uncertainty estimates. Table 4.2 shows the performance comparison of the MLP, CNN and LSTM models with different ranks of the approximations. Figure 4.5 contains analogous results for the ResNet-18 model. The results show that the post-training approximations of the mean field posterior covariance with ranks higher than one achieve predictive performance close to that of the mean field posterior with no approximations for all the analyzed model types, model sizes and layer types. Furthermore, Table 4.3 shows that, for the ResNet-18 model, the approximations with ranks higher than one also do not decrease the quality of the uncertainty estimates compared to the mean field posterior with no approximations[2]. These observations could

[2]We compute the Brier Score and the ECE using the implementations from the TensorFlow Probability (Dillon et al., 2017).

MNIST, MLP			
Method	-ELBO ↓	NLL ↓	Accuracy ↑
Mean-field	$0.431_{\pm 0.0057}$	$0.100_{\pm 0.0034}$	$97.6_{\pm 0.15}$
1-tied	$3.41_{\pm 0.019}$	$0.677_{\pm 0.0040}$	$93.6_{\pm 0.25}$
2-tied	$0.456_{\pm 0.0059}$	$0.107_{\pm 0.0033}$	$97.6_{\pm 0.15}$
3-tied	$0.450_{\pm 0.0059}$	$0.106_{\pm 0.0033}$	$97.6_{\pm 0.15}$

CIFAR100, CNN			
Method	-ELBO ↓	NLL ↓	Accuracy ↑
Mean-field	$3.83_{\pm 0.020}$	$2.23_{\pm 0.017}$	$42.1_{\pm 0.49}$
1-tied	$4.33_{\pm 0.021}$	$2.30_{\pm 0.016}$	$41.7_{\pm 0.49}$
2-tied	$3.88_{\pm 0.020}$	$2.24_{\pm 0.017}$	$42.2_{\pm 0.49}$
3-tied	$3.86_{\pm 0.020}$	$2.24_{\pm 0.017}$	$42.1_{\pm 0.49}$

IMDB, LSTM			
Method	-ELBO ↓	NLL ↓	Accuracy ↑
Mean-field	$0.536_{\pm 0.0058}$	$0.493_{\pm 0.0057}$	$80.1_{\pm 0.25}$
1-tied	$0.687_{\pm 0.0058}$	$0.491_{\pm 0.0056}$	$80.0_{\pm 0.25}$
2-tied	$0.621_{\pm 0.0058}$	$0.494_{\pm 0.0057}$	$80.1_{\pm 0.25}$
3-tied	$0.595_{\pm 0.0058}$	$0.493_{\pm 0.0056}$	$80.1_{\pm 0.25}$

Table 4.2: Impact of post-training low-rank approximation of the GMFVI-trained posterior standard deviation matrix on model's ELBO and predictive performance, for three types of models. We report mean and SEM of each metric across 100 weights samples. The low-rank approximations with ranks higher than one achieve predictive performance close to that of mean-field without the approximations.

be used as a form of post-training network compression. Moreover, they give rise to further interesting exploration directions such as formulating posteriors that exploit such a low-rank structure. In the next section we explore this particular direction while focusing on the first three model types (MLP, CNN, LSTM).

4.4 The k-tied Normal Distribution: Exploiting Low-Rank Parameter-Structure in Mean Field Posteriors

In the previous section, we have shown that it is possible to replace a reshaped diagonal matrix of posterior standard deviations, which is already trained using GMFVI, with its low-rank approximation without decreasing the predictive performance. In this section, we show that it is also possible to exploit this observation during training time. We achieve this by exploiting our novel variational family, the k-tied Normal distribution (see Section 4.3.1).

We show that using this distribution in the context of GMFVI in Bayesian neural networks

Method	Brier Score ↓	NLL ↓	ECE ↓
Mean-field	-0.761$_{\pm 0.0039}$	0.495$_{\pm 0.0080}$	0.0477
1-tied	-0.695 $_{\pm 0.0034}$	0.658$_{\pm 0.0069}$	0.1642
2-tied	-0.758$_{\pm 0.0038}$	0.503$_{\pm 0.0080}$	0.0540
3-tied	-0.758$_{\pm 0.0038}$	0.501$_{\pm 0.0079}$	0.0541

Table 4.3: Quality of predictive uncertainty estimates for the ResNet-18 model on the CIFAR10 dataset without and with post-training low-rank approximations of the GMFVI posterior standard deviation matrices in all the layers of the model. The approximations with ranks $k \geq 2$ match the quality of the predictive uncertainty estimates from the mean-field posteriors without the approximations. The quality of the predictive uncertainty estimates is measured by the negative log-likelihood (NLL), the Brier Score and the ECE (with 15 bins). For the NLL and the Brier Score metrics we report mean and SEM across 100 weights samples.

allows to reduce the number of network parameters, increase the signal-to-noise ratio of the stochastic gradient estimates and speed up model convergence, while maintaining the predictive performance of the standard parameterization of the GMFVI. We start by recalling the definition of the k-tied Normal distribution:

$$k\text{-}tied\text{-}\mathcal{N}(\mathbf{W}; \boldsymbol{\mu}_q, \mathbf{U}, \mathbf{V}) = \mathcal{N}\big(\boldsymbol{\mu}_q, \mathrm{diag}\big(\mathrm{vec}\big((\mathbf{U}\mathbf{V}^T)^2\big)\big)\big)$$

where the variational parameters are comprised of $\{\boldsymbol{\mu}_q, \mathbf{U}, \mathbf{V}\}$.

4.4.1 Experimental setting

We now introduce the experimental setting in which we evaluate the GMFVI variational posterior parameterized by the k-tied Normal distribution. We assess the impact of the described posterior in terms of predictive performance and reduction in the number of parameters for the same first three model types (MLP, CNN, LSTM) and respective datasets (MNIST, CIFAR-100, IMDB) as we used in the previous section. Additionally, we also analyze the impact of k-tied Normal posterior on the signal-to-noise ratio of stochastic gradient estimates of the variational lower bound for the CNN model as a representative example. Overall, the experimental setup is very similar to the one introduced in the previous section. Therefore, we highlight only the key differences here.

We apply the k-tied Normal variational posterior distribution to the same layers which we analyzed in the previous section. Namely, we use the k-tied Normal variational posterior for all the three layers of the MLP model, the two dense layers of the CNN model and the LSTM cell's kernel and recurrent kernel. We initialize the parameters u_{ik} from $\mathbf{U}$ and v_{jk} from $\mathbf{V}$ of the k-tied Normal distribution so that after the outer-product operation the respective standard deviations σ_{ij} have the same mean values as we obtain in the standard GMFVI posterior parameterization. In the experiments for this section, we use *KL annealing* (Sønderby et al., 2016), where we linearly scale-up the contribution of the $D_{\mathrm{KL}}[q_{\boldsymbol{\theta}}(\mathbf{w})||p(\mathbf{w})]$ term in Equation 2.4 from zero to its full contribution over the course of

training. Appendix A.3 describes the impact of KL annealing on the modelled uncertainty. Furthermore, additional details on the experimental setup are available in Appendix A.4.

4.4.2 Experimental results

We first investigate the predictive performance of the GMFVI Bayesian neural network models trained using the k-tied Normal posterior distribution, with different levels of tying k. We compare these results to those obtained from the same models, but trained using the standard parameterization of the GMFVI. Table 4.4 shows that for $k \geq 2$ the k-tied Normal posterior is able to achieve the performance competitive with the standard GMFVI posterior parameterization, while reducing the total number of model parameters. The benefits of using the k-tied Normal posterior are the most visible for models where the layers with the k-tied Normal posterior constitute a significant portion of the total number of the model parameters (e.g. the MLP model).

We further investigate the impact of the k-tied Normal posterior on the signal-to-noise ratio (SNR)[3] of stochastic gradient estimates of the variational lower bound (ELBO). In particular, we focus on the gradient SNR of the GMFVI posterior standard deviation parameters for which we perform the tying. These parameters are either u_{ik} and v_{jk} for the k-tied Normal posterior or σ_{ij} for the standard GMFVI parameterization, all optimized in their log forms for numerical stability. Table 4.5 shows that the u_{ik} and v_{jk} parameters used in the k-tied Normal posterior are trained with significantly higher gradient SNR than the σ_{ij} parameters used in the standard GMFVI parameterization. Consequently, Table 4.6 shows that the increased SNR from the k-tied Normal distribution translates into faster convergence for the MLP model, which uses the k-tied Normal distribution in all of its layers.

Note that the k-tied Normal posterior does not increase the training step time compared to the standard parameterization of the GMFVI, see Table 4.7 for the support of this claim[4]. Therefore, the k-tied Normal posterior speeds up model convergence also in terms of wall-clock time.

Figure 4.6 shows the convergence plots of validation negative ELBO for all the three model types. We observe that the impact of the k-tied Normal posterior on convergence depends on the model type. As shown in Table 4.6, the impact on the MLP model is strong and consistent with the k-tied Normal posterior increasing convergence speed compared to the standard GMFVI parameterization. For the LSTM model we also observe a similar speed-up. However, for the CNN model the impact of the k-Normal posterior on the ELBO convergence is much smaller. We hypothesize that this is due to the fact that we use the k-tied Normal posterior for all the layers trained using GMFVI in the MLP and the LSTM

[3]SNR for each gradient value is calculated as $E[g_b^2]/\text{Var}[g_b]$, where g_b is the gradient value for a single parameter. The expectation E and variance Var of the gradient values g_b are calculated over a window of last 10 batches.

[4]Code to compare the training step times of the k-tied Normal and the standard GMFVI is available under: `https://colab.research.google.com/drive/14pqe_VG5s49xlcXB-Jf8S9GoTFyjv4OF`. The code uses the network architecture from: `https://github.com/tensorflow/docs/blob/master/site/en/tutorials/keras/classification.ipynb`.

models, while in the CNN model we use the k-tied Normal posterior only for some of the GMFVI trained layers. More precisely, in the CNN model we use the k-tied Normal posterior only for the two dense layers, while the two convolutional layers are trained using the standard parameterization of the GMFVI.

Model & Dataset	Method	-ELBO ↓	NLL ↓	Accuracy ↑	#Par. [k] ↓
MNIST, MLP	Mean-field	$0.501_{\pm 0.0061}$	$0.133_{\pm 0.0040}$	$96.8_{\pm 0.18}$	957
MNIST, MLP	1-tied	$0.539_{\pm 0.0063}$	$0.155_{\pm 0.0043}$	$96.1_{\pm 0.19}$	482
MNIST, MLP	2-tied	$0.520_{\pm 0.0063}$	$0.129_{\pm 0.0039}$	$96.8_{\pm 0.18}$	484
MNIST, MLP	3-tied	$0.497_{\pm 0.0060}$	$0.120_{\pm 0.0038}$	$96.9_{\pm 0.18}$	486
CIFAR100, CNN	Mean-field	$3.72_{\pm 0.018}$	$2.16_{\pm 0.016}$	$43.9_{\pm 0.50}$	4,405
CIFAR100, CNN	1-tied	$3.65_{\pm 0.017}$	$2.12_{\pm 0.015}$	$45.5_{\pm 0.50}$	2,262
CIFAR100, CNN	2-tied	$3.76_{\pm 0.019}$	$2.15_{\pm 0.016}$	$44.3_{\pm 0.50}$	2,268
CIFAR100, CNN	3-tied	$3.73_{\pm 0.018}$	$2.13_{\pm 0.016}$	$44.3_{\pm 0.50}$	2,273
IMDB, LSTM	Mean-field	$0.538_{\pm 0.0054}$	$0.478_{\pm 0.0052}$	$79.5_{\pm 0.26}$	2,823
IMDB, LSTM	1-tied	$0.592_{\pm 0.0041}$	$0.512_{\pm 0.0040}$	$77.6_{\pm 0.26}$	2,693
IMDB, LSTM	2-tied	$0.560_{\pm 0.0042}$	$0.484_{\pm 0.0041}$	$78.2_{\pm 0.26}$	2,694
IMDB, LSTM	3-tied	$0.550_{\pm 0.0051}$	$0.491_{\pm 0.0050}$	$78.8_{\pm 0.26}$	2,695

Table 4.4: Impact of the k-tied Normal posterior on test ELBO, test predictive performance and number of model parameters. We report the test metrics on the test splits of the respective datasets as a mean and SEM across 100 weights samples after training each of the models for ≈300 epochs. The k-tied Normal distribution with rank $k \geq 2$ allows to train models with smaller number of parameters without decreasing the predictive performance.

Method	MNIST, MLP Dense 2, SNR at step		
	1000	5000	9000
Mean-field	$4.13_{\pm 0.027}$	$4.45_{\pm 0.091}$	$3.21_{\pm 0.035}$
1-tied	$5840_{\pm 190}$	$158_{\pm 3.8}$	$5.3_{\pm 0.20}$
2-tied	$7500_{\pm 240}$	$140_{\pm 11}$	$4.3_{\pm 0.26}$
3-tied	$7000_{\pm 270}$	$117_{\pm 1.7}$	$4.1_{\pm 0.20}$

Table 4.5: Mean gradient SNR in the log posterior standard deviation parameters of the Dense 2 layer of the MNIST MLP model at increasing training steps for different ranks of tying k. The k-tied Normal distribution significantly increases the SNR for these parameters. We observe a similar increase in the SNR from tying in all the layers that use the k-tied Normal posterior.

4.5 Related Work

The application of variational inference to neural networks dates back at least to Peterson (1987) and Hinton & Van Camp (1993a). Many developments[5] have followed those seminal

[5]We refer the interested readers to Zhang et al. (2018a) for a recent review of variational inference.

Method	MNIST, MLP, -ELBO at step		
	1000	5000	9000
Mean-field	$42.16_{\pm 0.070}$	$26.52_{\pm 0.016}$	$15.39_{\pm 0.016}$
1-tied	$43.11_{\pm 0.039}$	$14.85_{\pm 0.017}$	$2.06_{\pm 0.027}$
2-tied	$42.74_{\pm 0.090}$	$13.97_{\pm 0.023}$	$1.82_{\pm 0.017}$
3-tied	$42.63_{\pm 0.068}$	$13.61_{\pm 0.020}$	$1.80_{\pm 0.031}$

Table 4.6: Negative ELBO on the MNIST validation dataset at increasing training steps for different ranks of tying k. The higher SNR from the k-tied Normal posterior translates into the increased convergence speed for the MLP model. We report mean and SEM across 3 training runs with different random seeds in both the top right and the bottom right table.

Training method	Train step time [ms] ↓
Point estimate	$2.00_{\pm 0.0064}$
Standard GMFVI	$7.17_{\pm 0.014}$
2-tied Normal GMFVI	$6.14_{\pm 0.018}$

Table 4.7: Training step evaluation times for a simple model architecture with two dense layers for different training methods. We report mean and SEM of evaluation times across a single training run in the Google Colab environment linked in the footnote. The k-tied Normal posterior with $k = 2$ does not increase the train step evaluation times compared to the standard parameterization of the GMFVI posterior. We expect this to hold more generally because the biggest additional operation per step when using the k-tied Normal posterior is the $\mathbf{U}\mathbf{V}^T$ multiplication to materialize the matrix of posterior standard deviations $\mathbf{A}$, where $\mathbf{U} \in \mathbb{R}^{m \times k}$, $\mathbf{V} \in \mathbb{R}^{n \times k}$ and k is a small value (e.g., 2 or 3). The time complexity of this operations is $\mathcal{O}(kmn)$, which is usually negligible compared to the time complexity of data-weight matrix multiplication $\mathcal{O}(bmn)$, where b is the batch size.

research efforts, in particular regarding (1) the expressiveness of the variational posterior distribution and (2) the way the variational parameters themselves can be structured to lead to compact, easier-to-learn and scalable formulations. We organize the discussion of this section around those two aspects, with a specific focus on the Gaussian case.

Full Gaussian posterior. Because of their substantial memory and computational cost, Gaussian variational distributions with full covariance matrices have been primarily applied to (generalized) linear models and shallow neural networks (Jaakkola & Jordan, 1997; Barber & Bishop, 1998a; Marlin et al., 2011; Titsias & Lázaro-Gredilla, 2014; Miller et al., 2017; Ong et al., 2018).

To represent the dense covariance matrix efficiently in terms of variational parameters, several schemes have been proposed, including the sum of low-rank plus diagonal matrices (Barber & Bishop, 1998a; Seeger, 2000; Miller et al., 2017; Zhang et al., 2017; Ong et al., 2018), the Cholesky decomposition (Challis & Barber, 2011) or by operating instead on the precision matrix (Tan & Nott, 2018; Mishkin et al., 2018).

Gaussian posterior with block-structured covariances. In the context of Bayesian neural networks, the layers represent a natural structure to be exploited by the covariance matrix. When assuming independence across layers, the resulting covariance matrix exhibits a *block-diagonal structure* that has been shown to be a well-performing simplification of the dense setting (Sun et al., 2017; Zhang et al., 2017), with both memory and computational benefits.

Within each layer, the corresponding diagonal block of the covariance matrix can be represented by a Kronecker product of two smaller matrices (Louizos & Welling, 2016; Sun et al., 2017), possibly with a parameterization based on rotation matrices (Sun et al., 2017). Finally, using similar techniques, Zhang et al. (2017) proposed to use a block tridiagonal structure that better approximates the behavior of a dense covariance.

Fully factorized mean-field Gaussian posterior. A fully factorized Gaussian variational distribution constitutes the simplest option for variational inference. The resulting covariance matrix is diagonal and all underlying parameters are assumed to be independent. While the mean-field assumption is known to have some limitations—e.g., underestimated variance of the posterior distribution (Turner & Sahani, 2011) and robustness issues (Giordano et al., 2018)—it leads to scalable formulations, with already competitive performance, as for instance illustrated by the recent uncertainty quantification benchmark of Ovadia et al. (2019a).

Because of its simplicity and scalability, the fully-factorized Gaussian variational distribution has been widely used for Bayesian neural networks (Graves, 2011b; Ranganath et al., 2014; Blundell et al., 2015b; Hernández-Lobato & Adams, 2015; Zhang et al., 2017; Khan et al., 2018).

Our approach can be seen as an attempt to further reduce the number of parameters of the (already) diagonal covariance matrix. Closest to our approach is the work of Louizos & Welling (2016). Their matrix variate Gaussian distribution instantiated with the Kronecker product of the diagonal row- and column-covariance matrices leads to a rank-1 tying of the posterior variances. In contrast, we explore tying strategies beyond the rank-1 case, which we show to lead to better performance (both in terms of ELBO and predictive metrics). Importantly, we further prove that tying strategies with a rank greater than one cannot be represented in a matrix variate Gaussian distribution, thus clearly departing from Louizos & Welling (2016) (see Appendix A.1 for details).

Our approach can be also interpreted as a form of *hierarchical* variational inference from Ranganath et al. (2016). In this interpretation, the prior on the variational parameters corresponds to a Dirac distribution, non-zero only when a pre-specified low-rank tying relationship holds. More recently, Karaletsos et al. (2018) proposed a hierarchical structure which also couples network weights, but achieves this by introducing representations of network units as latent variables.

Our reduction in the number parameters through tying decreases the variance of the stochastic gradient estimates of the ELBO objective for the posterior standard deviation parameters. Alterantive methods for the variance reduction propose to either change the noise sampling scheme Kingma et al. (2015); Wen et al. (2018); Farquhar et al. (2020a) or to

determinize the variational posterior approximation Wu et al. (2019). Those methods can be combined with our method, but some of them are not valid in cases when our method is applicable (e.g. the method from Kingma et al. (2015) is not valid for convolutional layers).

We close this related work section by mentioning the existence of other strategies to produce more flexible approximate posteriors, e.g., normalizing flows (Rezende & Mohamed, 2015) and extensions thereof (Louizos & Welling, 2017).

4.6 Conclusion

In this chapter, we have shown that Bayesian Neural Networks trained with standard Gaussian Mean-Field Variational Inference learn posterior standard deviation matrices that can be approximated with little information loss by low-rank SVD decompositions. This suggests that richer parameterizations of the variational posterior may not always be needed, and that compact parameterizations can also work well. We used this insight to propose a simple, yet effective variational posterior parameterization, which speeds up training and reduces the number of variational parameters without degrading predictive performance on a range of model types.

In future work, we hope to scale up variational inference with compactly parameterized approximate posteriors to much larger models and more complex problems. For mean-field variational inference to work well in that setting several challenges will likely need to be addressed (Wenzel et al., 2020); improving the signal-to-noise ratio of ELBO gradients using our compact variational parameterizations may provide a piece of the puzzle.

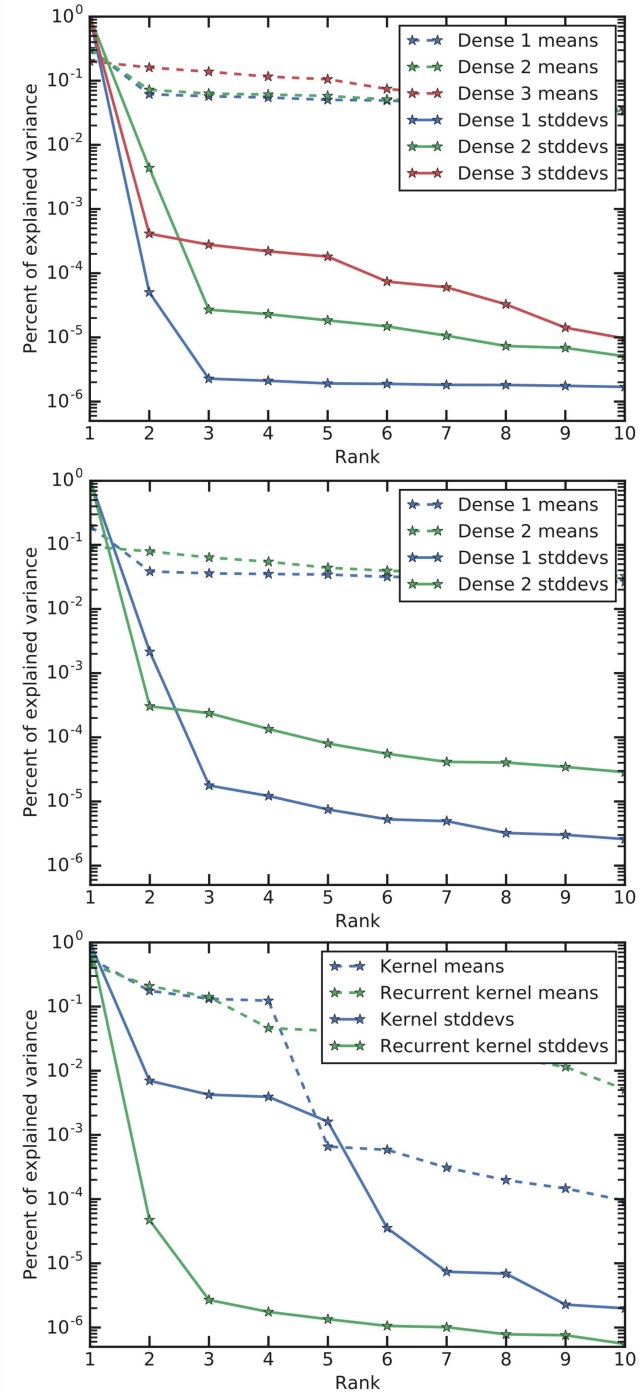


Figure 4.3: Fraction of variance explained per each singular value from SVD of matrices of posterior means and posterior standard deviations post-training in different dense layers of three model types trained using standard GMFVI: MLP (top), CNN (middle), LSTM (bottom). Unlike posterior means, posterior standard deviations clearly display strong low-rank structure, with most of the variance contained in the top few singular values.

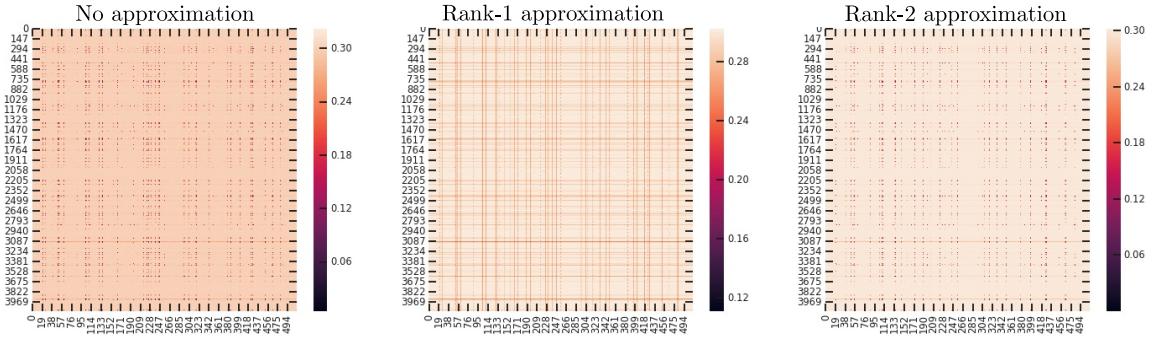
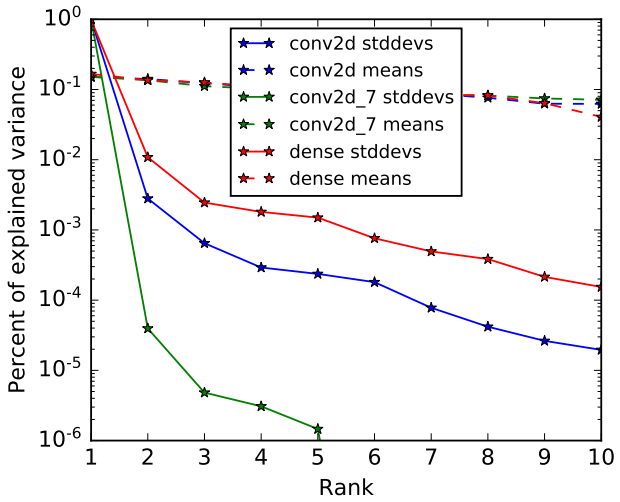


Figure 4.4: Post-training heat maps of the reshaped diagonal posterior standard deviation matrix for the first dense layer of a LeNet CNN trained using GMFVI on the CIFAR-100 dataset. Unlike the rank-1 approximation, the rank-2 approximation already looks visually very similar to the matrix with no approximation. This is consistent with the quantitative results from Figure 4.3, where the first two singular values explain most of the variance in the reshaped diagonal posterior standard deviation matrix.



Method	-ELBO ↓	NLL ↓	Accuracy ↑
Mean-field	$122.61_{\pm 0.012}$	$0.495_{\pm 0.0080}$	$83.5_{\pm 0.37}$
1-tied	$122.57_{\pm 0.012}$	$0.658_{\pm 0.0069}$	$81.7_{\pm 0.39}$
2-tied	$122.77_{\pm 0.012}$	$0.503_{\pm 0.0080}$	$83.2_{\pm 0.37}$
3-tied	$122.67_{\pm 0.012}$	$0.501_{\pm 0.0079}$	$83.2_{\pm 0.37}$

Figure 4.5: Unlike posterior means, the posterior standard deviations of both dense and convolutional layers in the ResNet-18 model trained using standard GMFVI display strong low-rank structure post-training and can be approximated without loss in predictive metrics. Top: Fraction of variance explained per each singular value of the matrices of converged posterior means and standard deviations. Bottom: Impact of post-training low-rank approximation of the posterior standard deviation matrices on the model's performance. We report mean and SEM of each metric across 100 weights samples.

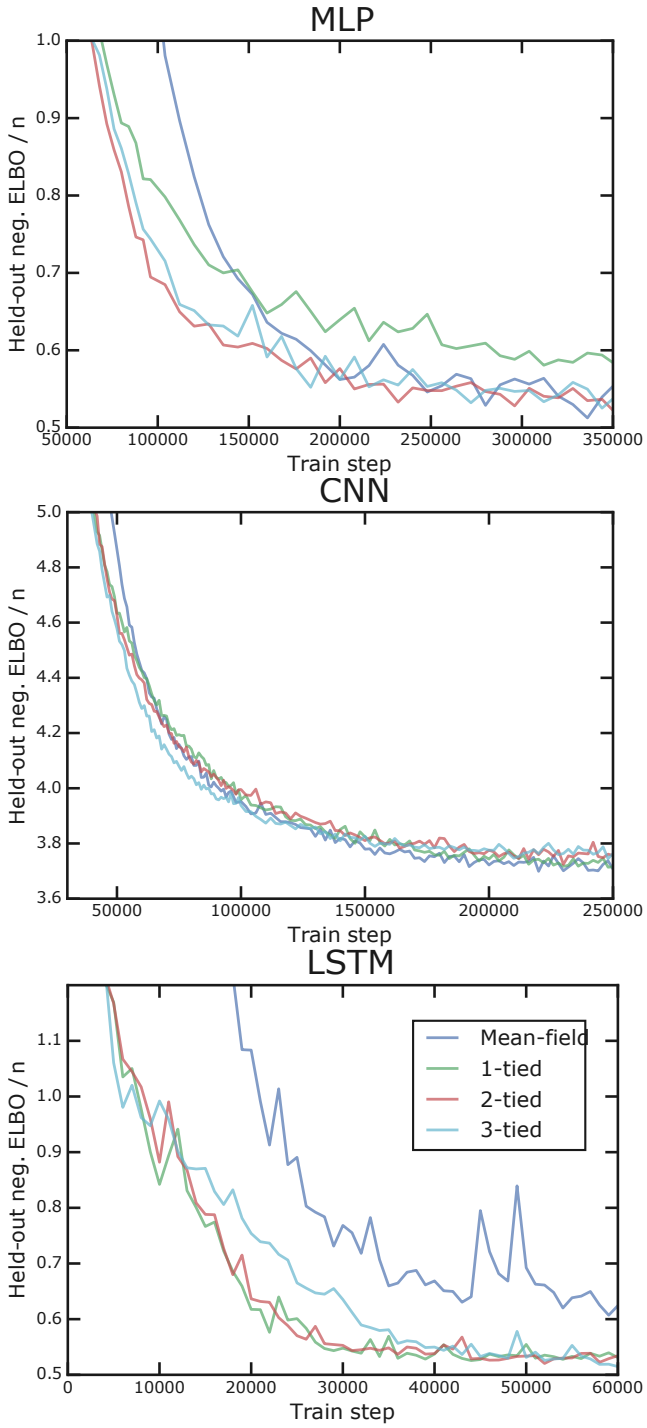


Figure 4.6: Convergence of negative ELBO (lower is better) for the three model types on their respective validation datasets when training with mean-field and k-tied variational posteriors. The k-tied Normal posteriors result in faster initial convergence for the MLP and LSTM models. For the CNN models the speed-up is not as significant when using the k-tied Normal posterior only for the two dense layers of the LeNet architecture.

Chapter 5

How Good is the Bayes Posterior in Deep Neural Networks Really?

5.1 Overview

During the past five years the Bayesian deep learning community has developed increasingly accurate and efficient approximate inference procedures that allow for Bayesian inference in deep neural networks. However, despite this algorithmic progress and the promise of improved uncertainty quantification and sample efficiency there are—as of early 2020—no publicized deployments of Bayesian neural networks in industrial practice. In this chapter, we cast doubt on the current understanding of Bayes posteriors in popular deep neural networks: we demonstrate through careful MCMC sampling that the posterior predictive induced by the Bayes posterior yields systematically worse predictions compared to simpler methods including point estimates obtained from SGD. Furthermore, we demonstrate that predictive performance is improved significantly through the use of a "cold posterior" that overcounts evidence. Such cold posteriors sharply deviate from the Bayesian paradigm but are commonly used as heuristic in Bayesian deep learning papers. We put forward several hypotheses that could explain cold posteriors and evaluate the hypotheses through experiments. Our work questions the goal of accurate posterior approximations in Bayesian deep learning: If the true Bayes posterior is poor, what is the use of more accurate approximations? Instead, we argue that it is timely to focus on understanding the origin of the improved performance of cold posteriors.

5.2 Introduction

We introduced the Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC) Bayesian neural networks that are the topic of this chapter in Section 2.3.2 of this thesis. Specifically, this chapter studies a surprising effect in SG-MCMC Bayesian neural networks shown in Figure 5.1, the "*Cold Posteriors*" effect: for deep neural networks the Bayes posterior (at

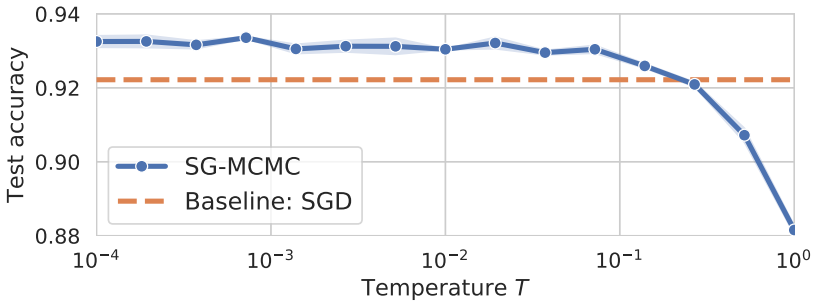


Figure 5.1: The "**cold posterior**" effect: for a ResNet-20 on CIFAR-10 we can improve the generalization performance significantly by cooling the posterior with a temperature $T \ll 1$, deviating from the Bayes posterior $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-U(\boldsymbol{\theta})/T)$ at $T = 1$.

temperature $T = 1$) works poorly but by cooling the posterior using a temperature $T < 1$ we can significantly improve the prediction performance. This empirical observation conflicts with the theoretical understanding of Bayes posteriors in deep neural networks, for which $T = 1$ should be optimal, as we explain next.

Cold Posteriors: among all temperized posteriors the best posterior predictive performance on holdout data is achieved at temperature $T < 1$.

5.2.1 Why Should Bayes ($T = 1$) be Better?

Why would we expect that predictions made by the *ensemble model* (2.2), could improve over predictions made at a single well-chosen parameter? There are three reasons: 1. *Theory*: for several models where the predictive performance can be analyzed it is known that the posterior predictive (2.2) can dominate common point-wise estimators based on the likelihood, (Komaki, 1996), even in the case of misspecification, (Fushiki et al., 2005; Ramamoorthi et al., 2015); 2. *Classical empirical evidence*: for classical statistical models, averaged predictions (2.2) have been observed to be more robust in practice, (Geisser, 1993); and 3. *Model averaging*: recent deep learning models based on deterministic model averages, (Lakshminarayanan et al., 2017; Ovadia et al., 2019b), have shown good predictive performance.

Note that a large body of work in the area of Bayesian deep learning in recent years is motivated by the assertion that predicting using (2.2) is desirable. We will confront this assertion through a simple experiment to show that our understanding of the Bayes posterior in deep models is limited. Our work makes the following **contributions**:

- We demonstrate for two models and tasks (ResNet-20 on CIFAR-10 and CNN-LSTM on IMDB) that the Bayes posterior predictive has poor performance compared to SGD-trained models.
- We put forth and systematically examine hypotheses that could explain the observed behaviour.
- We introduce two new diagnostic tools for assessing the approximation quality of stochastic gradient Markov chain Monte Carlo methods (SG-MCMC) and demonstrate

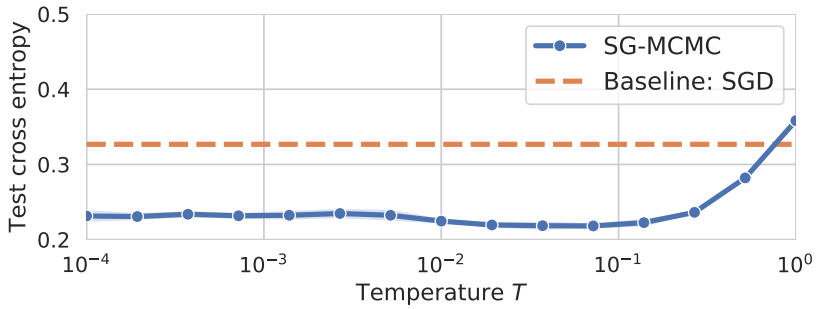


Figure 5.2: Predictive performance on the CIFAR-10 test set for a cooled ResNet-20 Bayes posterior. The SGD baseline is separately tuned for the same model (Appendix B.1.2).

that the posterior is accurately simulated by existing SG-MCMC methods.

5.3 Cold Posteriors Perform Better

We now examine the quality of the posterior predictive for two simple deep neural networks. We describe details of the models, priors, and approximate inference methods in Appendix B.1.1 to B.1.3. In particular, we will study the accuracy of our approximate inference and the influence of the prior in great detail in Section 5.5 and Section 5.6.2, respectively. Here we show that temperized Bayes ensembles obtained via low temperatures $T < 1$ outperform the true Bayes posterior at temperature $T = 1$.

5.3.1 Deep Learning Models: ResNet-20 and LSTM

ResNet-20 on CIFAR-10. Figure 5.1 and 5.2 show the test accuracy and test cross-entropy of a Bayes prediction (2.2) for a ResNet-20 on the CIFAR-10 classification task.[1] We can clearly see that both accuracy and cross-entropy are significantly improved for a temperature $T < 1/10$ and that this trend is consistent. Also, surprisingly this trend holds all the way to small $T = 10^{-4}$: the test performance obtained from an ensemble of models at temperature $T = 10^{-4}$ is superior to the one obtained from $T = 1$ and better than the performance of a single model trained with SGD. In Appendix B.7 we show that the uncertainty metrics Brier score Brier (1950) and expected calibration error (ECE) Naeini et al. (2015) are also improved by cold posteriors.

CNN-LSTM on IMDB text classification. Figure 5.3 shows the test accuracy and test cross-entropy of the tempered prediction (2.2) for a CNN-LSTM model on the IMDB sentiment classification task. The optimal predictive performance is again achieved for a tempered posterior with a temperature range of approximately $0.01 < T < 0.2$.

[1] A similar plot is Figure 3 in (Baldock & Marzari, 2019) and another is in the appendix of (Zhang et al., 2019a).

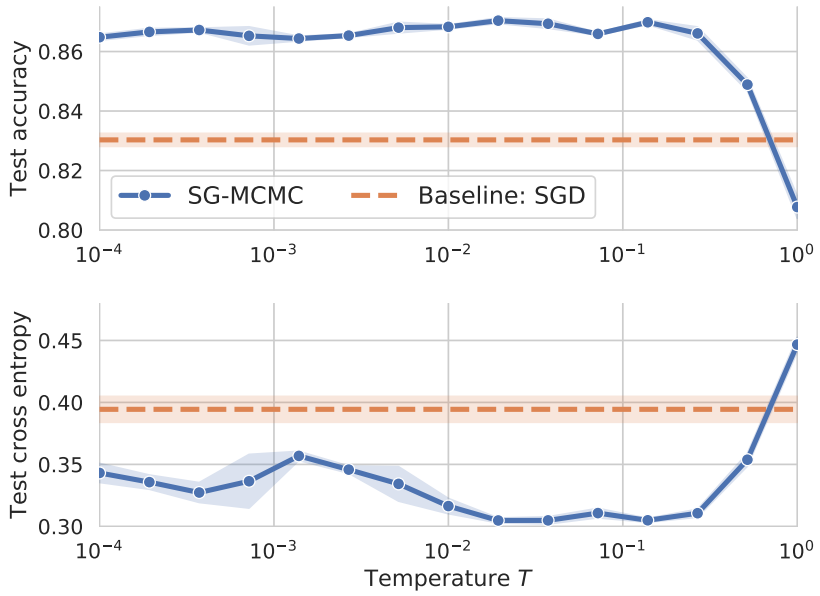


Figure 5.3: Predictive performance on the IMDB sentiment task test set for a tempered CNN-LSTM Bayes posterior. Error bars are ± one standard error over three runs. See Appendix B.1.4.

5.3.2 Why is a Temperature of $T < 1$ a Problem?

There are two reasons why cold posteriors are problematic. *First*, $T < 1$ corresponds to artificially sharpening the posterior, which can be interpreted as overcounting the data by a factor of $1/T$ and a rescaling[2] of the prior as $p(\theta)^{\frac{1}{T}}$. This is equivalent to a Bayes posterior obtained from a dataset consisting of $1/T$ replications of the original data, giving too strong evidence to individual models. For $T = 0$, all posterior probability mass is concentrated on the set of maximum a posteriori (MAP) point estimates. *Second*, $T = 1$ corresponds to the true Bayes posterior and performance gains for $T < 1$ point to a deeper and potentially resolvable problem with the prior, likelihood, or inference procedure.

5.3.3 Confirmation from the Literature

Should the strong performance of tempering the posterior with $T \ll 1$ surprise us? It certainly is an observation that needs to be explained, but it is not new: if we comb the literature of Bayesian inference in deep neural networks we find broader evidence for this phenomenon.

[2]E.g., using a Normal prior with temperature T results in a Normal distribution with scaled variance by a factor of T.

Related work that uses $T < 1$ posteriors in SG-MCMC. The following table lists work that uses SG-MCMC on deep neural networks and tempers the posterior.[3]

Reference	Temperature T
(Li et al., 2016)	$1/\sqrt{n}$
(Leimkuhler et al., 2019)	$T < 10^{-3}$
(Heek & Kalchbrenner, 2019)	$T = 1/5$
(Zhang et al., 2019a)	$T = 1/\sqrt{50000}$

Related work that uses $T < 1$ posteriors in Variational Bayes. In the variational Bayes approach to Bayesian neural networks, (Blundell et al., 2015b; Hinton & Van Camp, 1993b; MacKay et al., 1995; Barber & Bishop, 1998b) we optimize the parameters τ of a variational distribution $q(\boldsymbol{\theta}|\tau)$ by maximizing the evidence lower bound (ELBO),

$$\mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\tau)}\left[\sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta})\right] - \lambda D_{\mathrm{KL}}(q(\boldsymbol{\theta}|\tau)\|p(\boldsymbol{\theta})). \tag{5.1}$$

For $\lambda = 1$ this directly minimizes $D_{\mathrm{KL}}(q(\boldsymbol{\theta}|\tau) \,\|\, p(\boldsymbol{\theta}|\mathcal{D}))$ and thus for sufficiently rich variational families will closely approximate the true Bayes posterior $p(\boldsymbol{\theta}|\mathcal{D})$. However, in practice researchers discovered that using values $\lambda < 1$ provides better predictive performance, with common values shown in the following table.[4]

Reference	KL term weight λ in (5.1)
(Zhang et al., 2018b)	$\lambda \in \{1/2, 1/10\}$
(Bae et al., 2018)	tuning of λ, unspecified
(Osawa et al., 2019a)	$\lambda \in \{1/5, 1/10\}$
(Ashukha et al., 2020)	λ from 10^{-5} to 10^{-3}

In Appendix B.5 we show that the KL-weighted ELBO (5.1) arises from tempering the likelihood part of the posterior.

From the above list we can see that the cold posterior problem has left a trail in the literature, and in fact we are not aware of *any* published work demonstrating well-performing Bayesian deep learning at temperature $T = 1$. We now give details on how we perform accurate Bayesian posterior inference in deep learning models.

[3]For (Li et al., 2016) the tempering with $T = 1/\sqrt{n}$ arises due to an implementation mistake. For (Heek & Kalchbrenner, 2019) we communicated with the authors, and tempering arises due to overcounting data by a factor of 5, approximately justified by data augmentation, corresponding to $T = 1/5$. For (Zhang et al., 2019a) the original implementation contains inadvertent tempering, however, the authors added a study of tempering in a revision.

[4]For (Osawa et al., 2019a) scaling with λ arises due to their use of a "data augmentation factor" $\rho \in \{5, 10\}$.

5.4 Accurate SG-MCMC Simulation

In this section we describe how we achieve efficient and accurate simulation of SG-MCMC Bayesian neural network posteriors based on Langevin dynamics introduced in Section 2.3.2 of this thesis. This section does not contain any major novel contribution but instead combines existing work.

In practice there are two sources of error when following the Langevine dynamics (2.17–2.19):

- *Minibatch noise*: $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta})$ is an unbiased estimate of $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ but contains additional estimation variance.
- *Discretization error*: we incur error by following a continuous-time path (2.14–2.15) using discrete steps (2.17–2.19).

We use two methods to reduce these errors: *preconditioning* and *cyclical time stepping*.

Layerwise Preconditioning. Preconditioning through a choice of matrix $\mathbf{M}$ is a common way to improve the behavior of optimization methods. Li et al. (2016) and Ma et al. (2015) proposed preconditioning for SG-MCMC methods, and in the context of molecular dynamics the use of a matrix $\mathbf{M}$ has a long tradition as well, (Leimkuhler & Matthews, 2016). Li's proposal is an adaptive preconditioner inspired by RMSprop, (Tieleman & Hinton, 2012). Unfortunately, using the discretized Langevin dynamics with a preconditioner $\mathbf{M}(\boldsymbol{\theta})$ that depends on $\boldsymbol{\theta}$ compromises the correctness of the dynamics.[5] We propose a simpler preconditioner that limits the frequency of adaptating $\mathbf{M}$: after a number of iterations we estimate a new preconditioner $\mathbf{M}$ using a small number of batches, say 32, but without updating any model parameters. This preconditioner then remains fixed for a number of iterations, for example, the number of iterations it takes to visit the training set once, i.e. one epoch. We found this strategy to be highly effective at improving simulation accuracy. For details, please see Appendix B.4.

Cyclical time stepping. The second method to improve simulation accuracy is to decrease the discretization step size h. Chen et al. (2015) studied the consequence of both minibatch noise and discretization error on simulation accuracy and showed that the overall simulation error goes to zero for $h \searrow 0$. While lowering the step size h to a small value would also make the method slow, recently Zhang et al. (2019a) propose to perform *cycles* of iterations $t = 1, 2, \ldots$ with a high-to-low step size schedule $h_0\, C(t)$ described by an initial step size h_0 and a function $C(t)$ that starts at $C(1) = 1$ and has $C(L) = 0$ for a cycle length of L iterations. Such cycles retain fast simulation speed in the beginning while accepting simulation error. Towards the end of each cycle however, a small step size ensures an accurate simulation. We use the cosine schedule from (Zhang et al., 2019a) for $C(t)$, see Appendix B.1.

We integrate these two techniques together into a practical SG-MCMC procedure, Algorithm 1. When no preconditioning and no cosine schedule is used ($\mathbf{M} = I$ and $C(t) = 1$ in all iterations) and $T(t) = 0$ this algorithm is equivalent to *Tensorflow*'s SGD with momentum (Appendix B.3).

[5] Li et al. (2016) derives the required correction term, which however is expensive to compute and omitted in practice.

Algorithm 1: Symplectic Euler Langevin scheme.

Function `SymEulerSGMCMC`($\tilde{G}$, $\boldsymbol{\theta}^{(0)}$, ℓ, β, n, T)

Input: $\tilde{G} : \Theta \to \mathbb{R}$ mean energy function estimate; $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$ initial parameter; $\ell > 0$ learning rate; $\beta \in [0, 1)$ momentum decay; n total training set size; $T(t) \geq 0$ temperature schedule

Output: Sequence $\boldsymbol{\theta}^{(t)}$, $t = 1, 2, \ldots$

$h_0 \leftarrow \sqrt{\ell/n}$ `// SDE time step`

$\gamma \leftarrow (1 - \beta)\sqrt{n/\ell}$ `// friction`

Sample $\mathbf{m}^{(0)} \sim \mathcal{N}_d(0, I_d)$

$\mathbf{M} \leftarrow I$ `// Initial M`

for $t = 1, 2, \ldots$ **do**

  if *new epoch* **then**

    $\mathbf{m}_c \leftarrow \mathbf{M}^{-1/2}\,\mathbf{m}^{(t-1)}$

    $\mathbf{M} \leftarrow$ `EstimateM`$(\tilde{G}, \boldsymbol{\theta}^{(t-1)})$

    $\mathbf{m}^{(t-1)} \leftarrow \mathbf{M}^{1/2}\,\mathbf{m}_c$

  $h \leftarrow C(t)\,h_0$ `// Cyclic modulation`

  Sample $\mathbf{R}^{(t)} \sim \mathcal{N}_d(0, I_d)$ `// noise`

  $\mathbf{m}^{(t)} \leftarrow (1 - h\gamma)\,\mathbf{m}^{(t-1)} - hn\nabla_{\boldsymbol{\theta}}\tilde{G}(\boldsymbol{\theta}^{(t-1)}) + \sqrt{2\gamma h T(t)}\,\mathbf{M}^{1/2}\,\mathbf{R}^{(t)}$

  $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + h\,\mathbf{M}^{-1}\mathbf{m}^{(t)}$

  if *end of cycle* **then**

    yield $\boldsymbol{\theta}^{(t)}$ `// Parameter sample`

Coming back to the Cold Posteriors effect, what could explain the poor performance at temperature $T = 1$? With our Bayesian hearts, there are only three possible areas to examine: the inference, the prior, or the likelihood function.

5.5 Inference: Is it Accurate?

Both the Bayes posterior and the cooled posteriors are all intractable. Moreover, it is plausible that the high-dimensional posterior landscape of a deep network may lead to difficult-to-simulate SDE dynamics (2.14–2.15). Our approximate SG-MCMC inference method further has to deal with minibatch noise and produces only a finite sample approximation to the predictive integral (2.2). Taken together, could the Cold Posteriors effect arise from a poor inference accuracy?

5.5.1 Hypothesis: Inaccurate SDE Simulation

Inaccurate SDE Simulation Hypothesis: the SDE (2.14–2.15) is poorly simulated.

To gain confidence that our SG-MCMC method simulates the posterior accurately, we introduce diagnostics that previously have not been used in the SG-MCMC context:

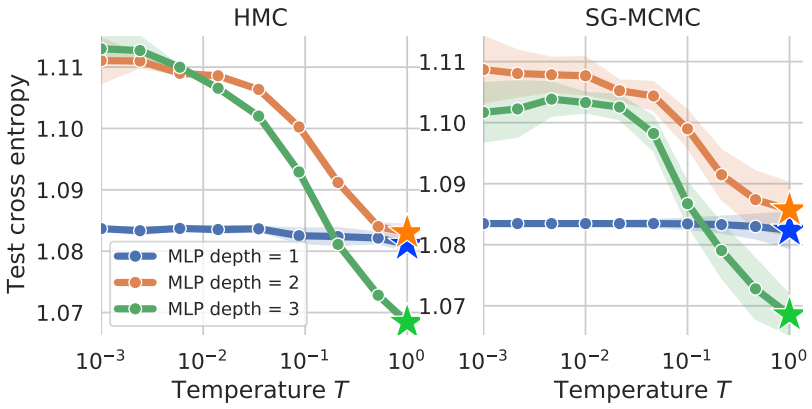


Figure 5.4: HMC (left) agrees closely with SG-MCMC (right) for synthetic data on multilayer perceptrons. A star indicates the optimal temperature for each model: for the synthetic data sampled from the prior there are no cold posteriors and both sampling methods perform best at $T = 1$.

- **Kinetic temperatures** (Appendix B.9.1): we report per-variable statistics derived from the moments $\mathbf{m}$. For these so called *kinetic temperatures* we know the exact sampling distribution under Langevin dynamics and compute their 99% confidence intervals.
- **Configurational temperatures** (Appendix B.9.2): we report per-variable statistics derived from $\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) \rangle$. For these *configurational temperatures* we know the expected value under Langevin dynamics.

We propose to use these diagnostics to assess simulation accuracy of SG-MCMC methods. We introduce the diagnostics and our new results in detail in Appendix B.9.

Inference Diagnostics Experiment: In Appendix B.10 we report a detailed study of simulation accuracy for both models. This study reports accurate simulation for both models when both preconditioning and cyclic time stepping are used. We can therefore with reasonably high confidence rule out a poor simulation of the SDE. All remaining experiments in this chapter also pass the simulation accuracy diagnostics.

5.5.2 Hypothesis: Biased SG-MCMC

Biased SG-MCMC Hypothesis: Lack of accept/reject Metropolis-Hastings corrections in SG-MCMC introduces bias.

In Markov chain Monte Carlo it is common to use an additional accept-reject step that corrects for bias in the sampling procedure. For MCMC applied to deep learning this correction step is too expensive and therefore omitted in SG-MCMC methods, which is valid for small time steps only, (Chen et al., 2015). If accept-reject is computationally feasible the resulting procedure is called *Hamiltonian Monte Carlo* (HMC) (Neal et al., 2011; Betancourt & Girolami, 2015; Duane et al., 1987; Hoffman & Gelman, 2014). Because it

provides unbiased simulation, we can consider HMC the *gold standard*, (Neal, 1995). We now compare gold standard HMC against SG-MCMC on a small example where comparison is feasible. We provide details of our HMC setup in Appendix B.15.

HMC Experiment: we construct a simple setup using a multilayer perceptron (MLP) where by construction $T = 1$ is optimal; such Bayes optimality must hold in expectation if the data is generated by the prior and model that we use for inference, (Berger, 1985). Thus, we can ensure that if the cold posterior effect is observed it must be due to a problem in our inference method. We perform all inference without minibatching ($|B| = n$) and test MLPs of varying number of one to three layers, ten hidden units each, and using the ReLU activation. As HMC implementation we use `tfp.mcmc.HamiltonianMonteCarlo` from *Tensorflow Probability* (Dillon et al., 2017; Lao et al., 2020): Details for our data and HMC are in Appendix B.14–B.15.

In Figure 5.4 the SG-MCMC results agree very well with the HMC results with optimal predictions at $T = 1$, i.e. no cold posteriors are present. For the cases tested we conclude that SG-MCMC is almost as accurate as HMC and the lack of accept-reject correction cannot explain cold posteriors. Appendix B.15 further shows that SG-MCMC and HMC are in good agreement when inspecting the KL divergence of their resulting predictive distributions.

5.5.3 Hypothesis: Stochastic Gradient Noise

Minibatch Noise Hypothesis: gradient noise from minibatching causes inaccurate sampling at $T = 1$.

Gradient noise due to minibatching can be heavy-tailed and non-Gaussian even for large batch sizes, (Simsekli et al., 2019). Our SG-MCMC method is only justified if the effect of noise will diminish for small time steps. We therefore study the influence of batch size on predictive performance through the following experiment.

Batchsize Experiment: we repeat the original ResNet-20/CIFAR-10 experiment at different temperatures for batch sizes in $\{32, 64, 128, 256\}$ and study the variation of the predictive performance as a function of batch size. Figure 5.5 and Figure 5.6 show that while there is a small variation between different batch sizes $T < 1$ remains optimal for all batch sizes. Therefore minibatch noise alone cannot explain the observed poor performance at $T = 1$.

For both ResNet and CNN-LSTM the best cross-entropy is achieved by the smallest batch size of 32 and 16, respectively. The smallest batch size has the *largest* gradient noise. We can interpret this noise as an additional heat source that increases the effective simulation temperature. However, the noise distribution arising from minibatching is anisotropic, (Zhu et al., 2019), and this could perhaps aid generalization. We will not study this hypothesis further here.

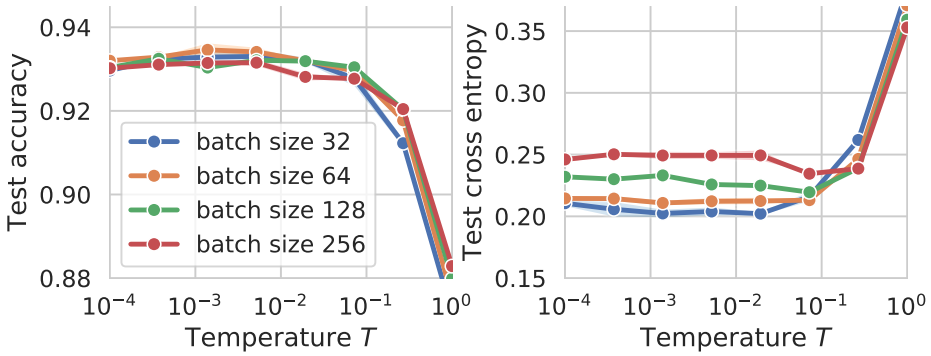


Figure 5.5: Batch size dependence of the ResNet-20/CIFAR-10 ensemble performance, reporting mean and standard error (3 runs): for all batch sizes the optimal predictions are obtained for $T < 1$.

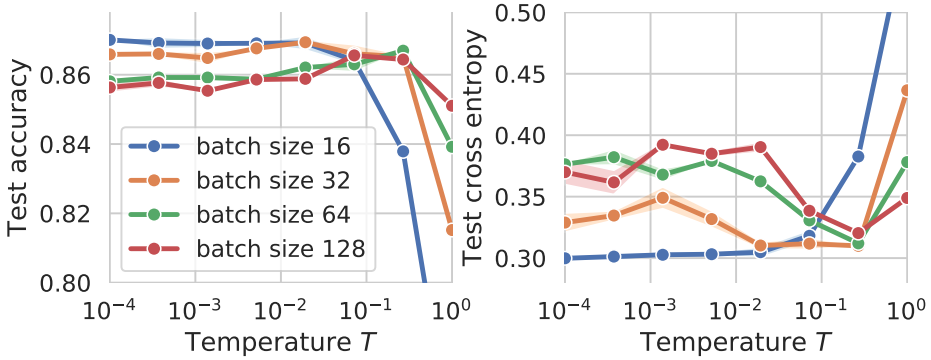


Figure 5.6: Batch size dependence of the CNN-LSTM/IMDB ensemble performance, reporting mean and standard error (3 runs): for all batch sizes, the optimal performance is achieved at $T < 1$.

5.5.4 Hypothesis: Bias-Variance Trade-off

Bias-variance Tradeoff Hypothesis: For $T = 1$ the posterior is diverse and there is high variance between model predictions. For $T \ll 1$ we sample nearby modes and reduce prediction variance but increase bias; the variance dominates the error and reducing variance ($T \ll 1$) improves predictive performance.

If this hypothesis were true then simply collecting more ensemble members, $S \to \infty$, would reduce the variance to arbitrary small values and thus fix the poor predictive performance we observe at $T = 1$. Doing so would require running our SG-MCMC schemes for longer—potentially for much longer. We study this question in detail in Appendix B.6 and conclude by an asymptotic analysis that the amount of variance cannot explain cold posteriors.

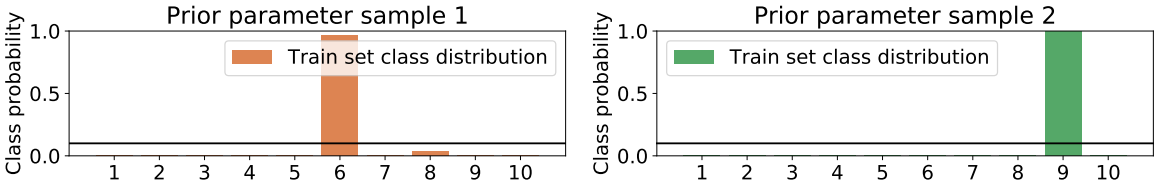


Figure 5.7: ResNet-20/CIFAR-10 typical prior predictive distributions for 10 classes under a $\mathcal{N}(0, I)$ prior averaged over the entire training set, $\mathbb{E}_{x\sim p(x)}[p(y|x, \boldsymbol{\theta}^{(i)})]$. Each plot is for one sample $\boldsymbol{\theta}^{(i)} \sim \mathcal{N}(0, I)$ from the prior. Given a sample $\boldsymbol{\theta}^{(i)}$ the average training data class distribution is highly concentrated around the same classes for all x.

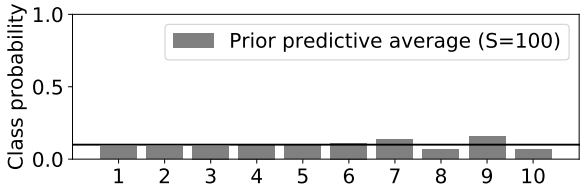


Figure 5.8: ResNet-20/CIFAR-10 prior predictive $\mathbb{E}_{x\sim p(x)}[\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta})}[p(y|x, \boldsymbol{\theta})]]$ over 10 classes, estimated using $S = 100$ prior samples $\boldsymbol{\theta}^{(i)}$ and all training images.

5.6 Why Could the Bayes Posterior be Poor?

With some confidence in our approximate inference procedure what are the remaining possibilities that could explain the cold posterior effect? The remaining two places to look at are the likelihood function and the prior.

5.6.1 Problems in the Likelihood Function?

For Bayesian deep learning we use the same likelihood function $p(y|x, \boldsymbol{\theta})$ as we use for SGD. Therefore, because the same likelihood function works well for SGD it appears an unlikely candidate to explain the cold posterior effect. However, current deep learning models use a number of techniques—such as data augmentation, dropout, and batch normalization—that are not formal likelihood functions. This observations brings us to the following hypothesis.

Dirty Likelihood Hypothesis: Deep learning practices that violate the likelihood principle (batch normalization, dropout, data augmentation) cause deviation from the Bayes posterior.

In Appendix B.11 we give a theory of "*Jensen posteriors*" which describes the likelihood-like functions arising from modern deep learning techniques. We report an experiment (Appendix B.11.4) that—while slightly inconclusive—demonstrates that cold posteriors remain when a clean likelihood is used in a suitably modified ResNet model; the CNN-LSTM model already had a clean likelihood function.

5.6.2 Problems with the Prior?

So far we have used a simple Normal prior, $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$, as was done in prior work (Zhang et al., 2019a; Heek & Kalchbrenner, 2019; Ding et al., 2014; Li et al., 2016; Zhang et al., 2018b). But is this a good prior?

One could hope, that perhaps with an informed and structured model architecture, a simple prior could be sufficient in placing prior beliefs on suitable functions, as argued by Wilson (2019). While plausible, we are mildly cautious because there are known examples where innocent looking priors have turned out to be unintentionally highly informative.[6] Therefore, with the cold posterior effect having a track record in the literature, perhaps $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$ could have similarly unintended effects of placing large prior mass on undesirable functions. This leads us to the next hypothesis.

Bad Prior Hypothesis: The current priors used for BNN parameters are inadequate, unintentionally informative, and their effect becomes stronger with increasing model depths and capacity.

To study the quality of our prior, we study typical functions obtained by sampling from the prior, as is good practice in model criticism, (Gelman et al., 2013).

Prior Predictive Experiment: for our ResNet-20 model we generate samples $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$ and look at the induced predictive distribution $\mathbb{E}_{x \sim p(x)}[p(y|x, \boldsymbol{\theta}^{(i)})]$ for each parameter sample, using the real CIFAR-10 training images. From Figure 5.7 we see that typical prior draws produce concentrated class distributions, indicating that the $\mathcal{N}(0, I)$ distribution is a poor prior for the ResNet-20 likelihood. From Figure 5.8 we can see that the average predictions obtained from such concentrated functions remain close to the uniform class distribution. Taken together, from a subjective Bayesian view $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$ is a *poor prior*: typical functions produced by this prior place a high probability the same few classes for all x. In Appendix B.12 we carry out another prior predictive study using He-scaling priors, (He et al., 2015b), which leads to similar results.

Prior Variance σ Scaling Experiment: in the previous experiment we found that the standard Normal prior is poor. Can the Normal prior $p(\boldsymbol{\theta}) = \mathcal{N}(0, \sigma)$ be fixed by using a more appropriate variance σ? For our ResNet-20 model we employ Normal priors of varying variances. Figure 5.12 shows that the cold posterior effect is present for all variances considered. Further investigations for known scaling laws in deep networks is given in Appendix B.12. The cold posterior effect cannot be resolved by using the right scaling of the Normal prior.

Training Set Size n Scaling Experiment: the posterior energy $U(\boldsymbol{\theta})$ in (2.12) sums over all n data log-likelihoods but adds $\log p(\boldsymbol{\theta})$ only once. This means that the influence of $\log p(\boldsymbol{\theta})$ vanishes at a rate of $1/n$ and thus the prior will exert its strongest influence for small n. We now study what happens for small n by comparing the Bayes predictive under

[6] A shocking example in the Dirichlet-Multinomial model is given by Nemenman et al. (2002). Importantly the unintended effect of the prior was not recognized when the model was originally proposed by Wolpert & Wolf (1995).

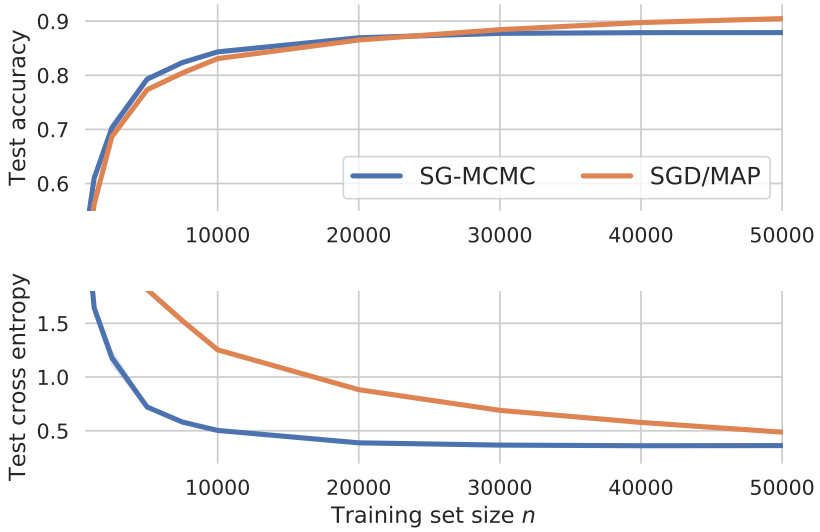


Figure 5.9: ResNet-20/CIFAR-10 predictive performance as a function of training set size n. The Bayes posterior ($T = 1$) degrades gracefully as n decreases, whereas SGD/MAP performs worse.

a $\mathcal{N}(0, I)$ prior against performing SGD maximum a posteriori (MAP) estimation on the *same* log-posterior.[7]

Figure 5.9 and Figure 5.10 show the predictive performance for ResNet-20 on CIFAR-10 and CNN-LSTM on IMDB, respectively. These results differ markedly between the two models and datasets: for ResNet-20 / CIFAR-10 the Bayes posterior at $T = 1$ degrades gracefully for small n, whereas SGD suffers large losses in test cross-entropy for small n. For CNN-LSTM / IMDB predictions from the Bayes posterior at $T = 1$ deteriorate quickly in both test accuracy and cross entropy. In all these runs SG-MCMC and SGD/MAP work with the same $U(\boldsymbol{\theta})$ and the difference is between integration and optimization. The results are inconclusive but somewhat implicate the prior in the cold posterior effect: as n becomes small there is an increasing difference between the cross-entropy achieved by the Bayes prediction and the SGD estimate, for large n the SGD estimate performs better.

Capacity Experiment: we consider a MLP using a $\mathcal{N}(0, I)$ prior and study the relation of the network capacity to the cold posterior effect. We train MLPs of varying depth (number of layers) and width (number of units per layer) at different temperatures on CIFAR-10. Figure 5.11 shows that for increasing capacity the cold posterior effect becomes more prominent. This indicates a connection between model capacity and strength of the cold posterior effect.

[7] For SGD we minimize $U(\boldsymbol{\theta})/n$.

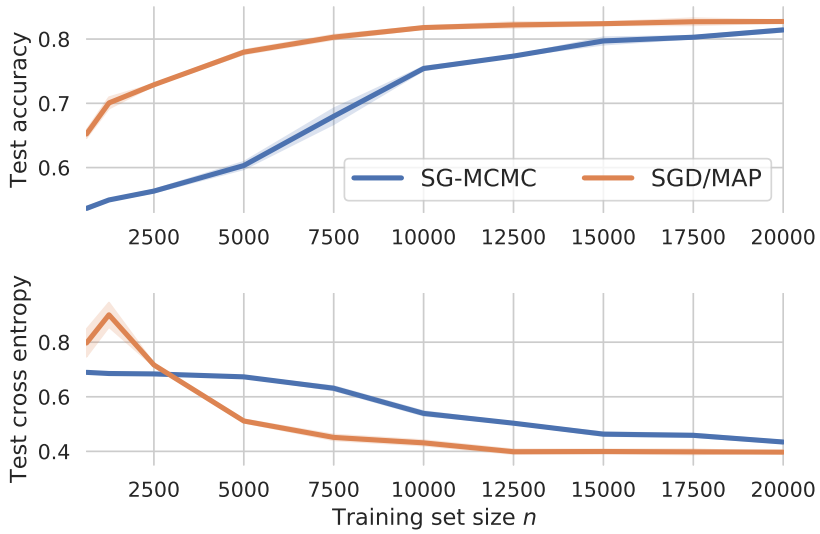


Figure 5.10: CNN-LSTM/IMDB predictive performance as a function of training set size n. The Bayes posterior ($T = 1$) suffers more than the SGD performance, indicating a problematic prior.

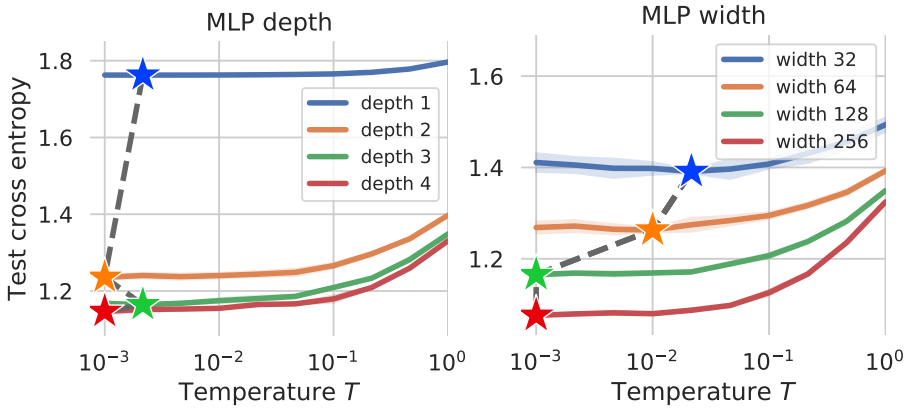


Figure 5.11: MLP of different capacities (depth and width) on CIFAR-10. Left: we fix the width to 128 and vary the depth. Right: we fix the depth to 3 and vary the width. Increasing capacity lowers the optimal temperature.

5.6.3 Inductive Bias due to SGD?

Implicit Initialization Prior in SGD: The inductive bias from initialization is strong and beneficial for SGD but harmed by SG-MCMC sampling.

Optimizing neural networks via SGD with a suitable initialization is known to have a beneficial inductive bias leading to good local optima, (Masters & Luschi, 2018; Mandt et al., 2017). Does SG-MCMC perform worse due to decreasing the influence of that bias? We address this question by the following experiment. We first run SGD until convergence, then

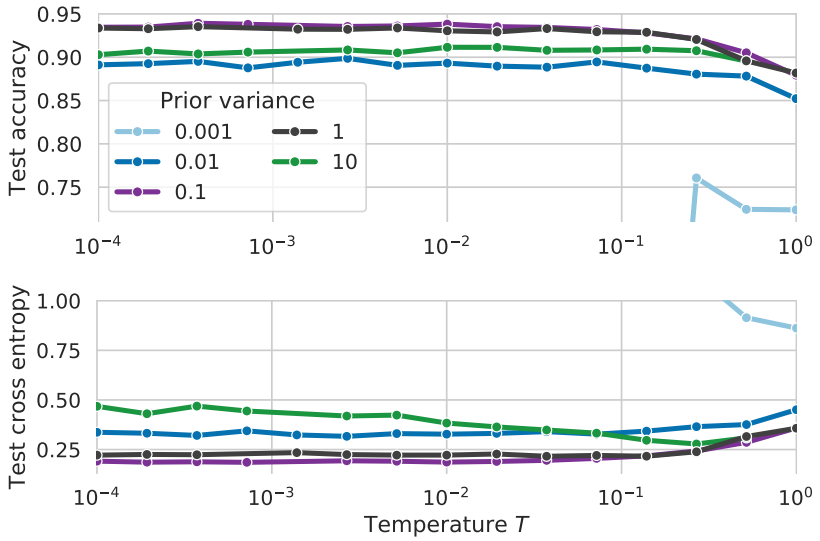


Figure 5.12: ResNet-20/CIFAR-10 predictive performance as a function of temperature T for different priors $p(\boldsymbol{\theta}) = \mathcal{N}(0, \sigma)$. The cold posterior effect is present for all choices of the prior variance σ. For all models the optimal temperature is significantly smaller than one and for $\sigma = 0.001$ the performance is poor for all temperatures. There is no "simple" fix of the prior.

switch over to SG-MCMC sampling for 500 epochs (10 cycles), and finally switch back to SGD again. Figure 5.13 shows that SGD initialized by the last model of the SG-MCMC sampling dynamics recovers the same performance as vanilla SGD. This indicates that the beneficial initialization bias for SGD is not destroyed by SG-MCMC. Details can be found in Appendix B.8.

5.7 Alternative Explanations?

Are there other explanations we have not studied in this chapter?

Masegosa Posteriors. A compelling analysis of the failure to predict well under the Bayes posterior is given by Masegosa (2019). In his analysis he first follows Germain et al. (2016) in identifying the Bayes posterior as a solution of a loose PAC-Bayes generalization bound on the predictive cross-entropy. He then uses recent results demonstrating improved Jensen inequalities, (Liao & Berg, 2019), to derive alternative posteriors. These alternative posteriors are *not* Bayes posteriors and in fact explicitly encourage diversity among ensemble member predictions. Moreover, the alternative posteriors can be shown to dominate the predictive performance achieved by the Bayes posterior when the model is misspecified. We believe that these new "Masegosa-posteriors", while not explaining cold posteriors fully, may provide a more desirable approximation target than the Bayes posterior. In addition, the Masegosa-posterior is compatible with both variational and SG-MCMC type algorithms.

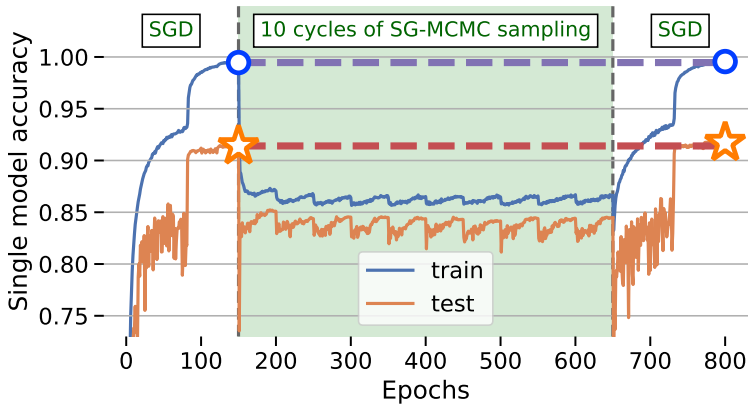


Figure 5.13: Do the SG-MCMC dynamics harm a beneficial initialization bias used by SGD? We first train a ResNet-20 on CIFAR-10 via SGD, then switch over to SG-MCMC sampling and finally switch back to SGD optimization. We report the single-model test accuracy of SGD and the SG-MCMC chain as function of epochs. SGD recovers from being initialized by the SG-MCMC state.

Tempered observation model? In (Wilson & Izmailov, 2020, Section 8.3) it is claimed that cold posteriors in one model correspond to untempered ($T = 1$) Bayes posteriors in a modified model by a simple change of the likelihood function. If this were the case, this would resolve the cold posterior problem and in fact point to a systematic way how to improve the Bayes posterior in many models. However, the argument in (Wilson & Izmailov, 2020) is wrong, which we demonstrate and discuss in detail in Appendix B.13.

5.8 Related Work on Tempered Posteriors

Statisticians have studied *tempered* or *fractional* posteriors for $T > 1$. Motivated by the behavior of Bayesian inference in *misspecified* models (Grünwald et al., 2017; Jansen, 2013) develop the *SafeBayes* approach and Bhattacharya et al. (2019) develops *fractional posteriors* with the goal of slowing posterior concentration. The use of multiple temperatures $T > 1$ is also common in Monte Carlo simulation in the presence of rough energy landscapes, e.g. (Earl & Deem, 2005; Sugita & Okamoto, 1999; Swendsen & Wang, 1986). However, the purpose of such tempering is to aid in accurate sampling at a desired target temperature, but not in changing the target distribution. Mandt et al. (2016) studies temperature as a latent variable in the context of variational inference and shows that models often select temperatures different from one.

5.9 Conclusion

This chapter has raised the question of cold posteriors but we did not fully resolve nor fix the cause for the cold posterior phenomenon. Yet our experiments suggest the following.

SG-MCMC is accurate enough: our experiments (Section 5.5–5.6) and novel diagnostics (Appendix B.9) indicate that current SG-MCMC methods are robust, scalable, and accurate enough to provide good approximations to parameter posteriors in deep nets.

Cold posteriors work: while we do not fully understand cold posteriors, tempered SG-MCMC ensembles provide a way to train ensemble models with improved predictions compared to individual models. However, taking into account the added computation from evaluating ensembles, there may be more practical methods, (Lakshminarayanan et al., 2017; Wen et al., 2019; Ashukha et al., 2020).

More work on priors for deep nets is needed: the experiments in Section 5.6.2 implicate the prior $p(\boldsymbol{\theta})$ in the cold posterior effect, although the prior may not be the only cause. Our investigations fail to produce a "simple" fix based on scaling the prior variance appropriately. Future work on suitable priors for Bayesian neural networks is needed, building on recent advances, (Sun et al., 2019; Pearce et al., 2019; Flam-Shepherd et al., 2017; Hafner et al., 2018).

Acknowledgements. We would like to thank Dustin Tran for reading multiple drafts and providing detailed feedback on the work. We also thank the four anonymous ICML 2020 reviewers for their detailed and helpful feedback.

Chapter 6

Cross-lingual Prosody Transfer for Expressive Machine Dubbing

6.1 Overview

Prosody transfer is well-studied in the context of expressive speech synthesis. Cross-lingual prosody transfer, however, is challenging and has been under-explored to date. In this chapter, we present a novel solution to learn prosody representations that are transferable across languages and speakers for machine dubbing of expressive multimedia contents. Multimedia contents often contain field recordings. To enable prosody transfer from noisy audios, we introduce a novel noise modelling module that disentangles noise conditioning from prosody conditioning, and thereby gains independent control of noise levels in the synthesised speech. We augment noisy training data with clean data to improve the ability of the model to map the denoised reference audio to clean speech. Our proposed system can generate speech with context-matching prosody and closes the gap between a strong baseline and human expressive dialogs by 11.2%

6.2 Introduction

Intonation, stress, rhythm and style are factors of speech that are collectively referred to as prosody. To study and apply these factors for the purpose of speech generation, various acoustically inspired labelling schemes have been designed. In Van Coile et al. (1994), the transplantation of prosody from an original speech clip via a system called PROTRAN was proposed. This technique involves an encoding of stylized pitch contours and phoneme durations into a low bit-rate *enriched phonetic transcription* that can be used in conjunction with desired text to reproduce the prosody of an original recording. In this work, we circumvent the labour-intensive schematizing and labelling of prosody. We adopt the term *prosody* as a general term that constitutes learned latent representations from ground truth speech audios. Similar to the definition in Skerry-Ryan et al. (2018b), prosody in this chapter

refers to the encoding of variations in speech signal that remains after accounting for the variations due to phonetics, language, speaker identity, and channel effects (i.e. the recording environment and ambient noise).

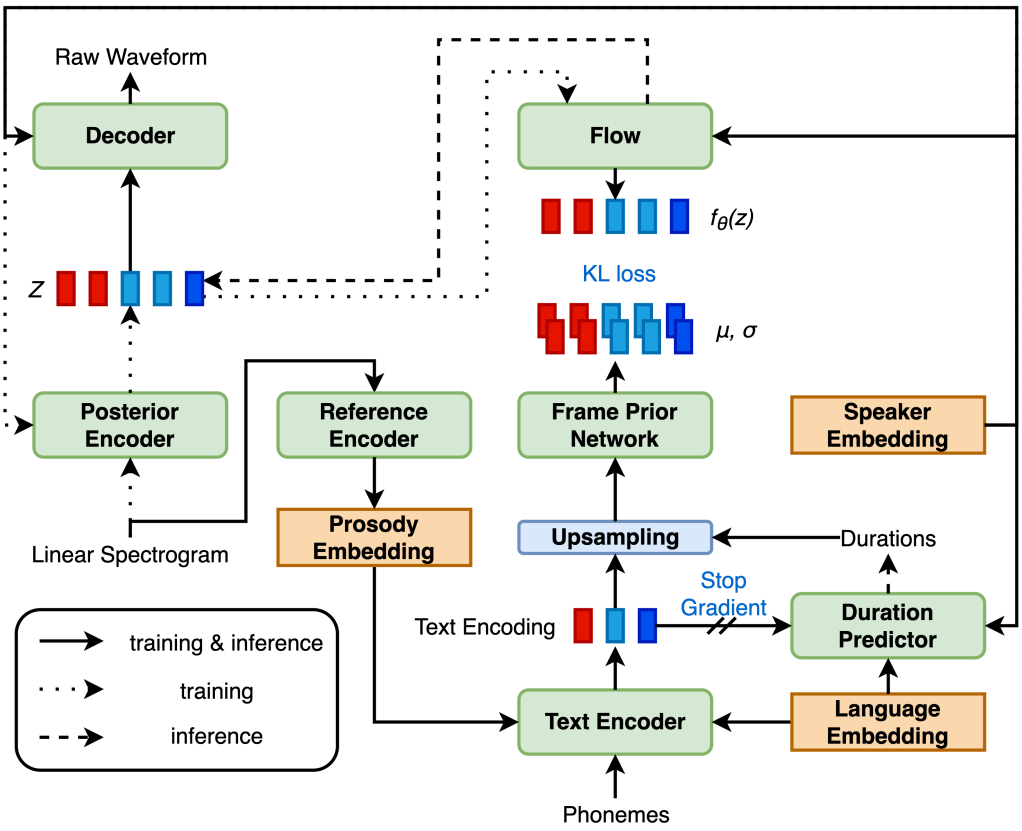


Figure 6.1: Architecture diagram of the VIPT system with prosody reference encoder.

In this chapter, we focus on cross-lingual speech synthesis for machine dubbing applications described in Section 3.5 of this thesis. Existing speech synthesis methods in machine dubbing (Federico et al., 2020; Matoušek & Vít, 2012; Effendi et al., 2022b) generate speech only based on translated text, but do not model nor transfer the expression of corresponding speech in the original language. For machine dubbing of expressive multimedia contents such as videos from various sources, it is important to convey the same emotion and expression as in the original speech (Brannon et al., 2021). Here, we explore cross-lingual prosody transfer for expressive speech synthesis of multimedia contents. We define cross-lingual prosody transfer as the transfer of prosody representations from speech in a source language from a source speaker to generate speech in a target language with voice characteristics of a target speaker. While exact prosody delivery varies across languages, the prosody of speech expressing the same emotions in related languages exhibits highly correlated prosody, as discussed in Section 4.6 in Brannon et al. (2021). Therefore, we explore these cross-lingual

correlations for the purpose of prosody transfer. We study European languages such as English, German, French, Italian and Spanish, and focus on English-Spanish prosody transfer. We do not focus here on more distant languages such as Japanese.

Cross-lingual prosody transfer brings additional context from speech in source language, but it also involves a number of challenges that are not present in conventional Text-To-Speech (TTS) solutions such as those introduced in Section 3.2 of this thesis. First, currently cross-lingual prosody transfer has to be learned without access to multilingual parallel speech datasets due to the scarcity of such datasets. The available parallel datasets (Jia et al., 2022) lack expressivity. The absence of expressive parallel datasets also means speech-to-speech translation methods (Lee et al., 2022) are not applicable. Second, available non-parallel speech datasets lack the full range of expressivity present in human speech (Wang et al., 2021). Therefore, we resort to gathering expressive speech of different languages and speakers from in-house multimedia data. However, such data was not recorded for the purpose of TTS systems. For example, the data contains channel noise, which needs to be alleviated to generate clean and expressive speech.

We propose a solution based on conditional variational autoencoder with adversarial learning for end-to-end text-to-speech (VITS) (Kim et al., 2021), which we described in Section 3.4.2. Our solution learns cross-lingual prosody transfer from non-parallel data. We use parallel translated texts during inference, but our proposed system doesn't require parallel text nor parallel audio data during training. It enables the cross-lingual prosody transfer by learning prosody representations that are agnostic to speakers and languages. The prosody representations are learnt via a variational reference encoder (Zhang et al., 2019c) with carefully balanced regularization, as introduced in Section 3.4.1. The learnt representations can be transplanted from a reference audio in source language spoken by a source speaker to generate speech in target language with the voice characteristics of a target speaker. Furthermore, to improve the robustness of our model to noisy reference audio, we propose two different approaches. The first approach utilizes a noise modelling extension to our reference encoder module that disentangles the prosody and the channel noise, where a denoised signal extracted from a reference audio is utilized. On the other hand, in the second approach, we augment the training data with clean speech data to improve the capability of our model to map a denoised reference audio to clean speech. Both approaches allow our system to learn from noisy data and to generate high-quality clean speech in a target language even when it is provided with a noisy reference audio from the source language.

Related to our system, numerous works on the prosody transfer within a single language have been proposed, such as with the use of reference encoder (Skerry-Ryan et al., 2018a), with style tokens (Wang et al., 2018), or with variational autoencoder (VAE) (Zhang et al., 2019c), as introduced in Section 3.4.1. Concurrent to us, Mitsui et al. (2022) also extended VITS with a reference encoder. The cross-lingual setting was explored in Rattcliffe et al. (2022a), but they focused on style transfer based on categorical labels, which are provided as ground truth during training. Last but not least, explicit noise modelling in TTS systems has also been studied (Zhang et al., 2021; Saeki et al., 2022), but transferring prosody from noisy reference recordings was not explored in these studies. To the best of our knowledge,

our method is the first to address the problem of cross-lingual prosody transfer for machine dubbing and is robust to noisy data. To sum up, our contributions are:

- We show that cross-lingual prosody transfer can be achieved with a multilingual model trained without parallel data.
- We propose a reference encoder architecture that disentangles prosody and channel noise allowing for clean speech synthesis from a noisy reference audio. We also investigate the augmentation of noisy data with clean training data to improve the capability of the model to map a denoised reference input to clean speech.

6.3 Modelling

Our proposed model consists of a backbone adapted from VITS (Kim et al., 2021), a prosody reference encoder to encode prosody information from the reference audio, and an optional noise reference encoder to model noise information. Figure 6.1 shows an overview of the proposed model architecture.

We derive our base model by adapting the following changes from the literature to the backbone VITS architecture that we briefly mentioned in Section 3.4.2. First, we replace VITS's monotonic alignment search algorithm with explicit duration predictor and extend prior encoder module with a frame prior network as in Zhang et al. (2022). Second, we incorporate speaker embeddings and language embeddings as in Cho et al. (2022) for training on multi-speaker and multi-lingual datasets. This is also depicted in Figure 6.1. Finally, we replace HiFiGAN decoder (Kong et al., 2020a) with a BigVGAN-base decoder (Lee et al., 2023) as BigVGAN shows improved generalization performance compared to HiFiGAN. We find that these changes significantly improve over the original VITS and keep them fixed in all our experiments.

6.3.1 Prosody Encoder

The prosody reference encoder extracts prosody embedding from a reference speech input as explained in Section 3.4.1. As we explicitly condition speaker and language variations via respective embeddings, the prosody encoder captures the remaining variability related to prosody. The prosody embedding is used to condition the model to synthesise speech with similar prosody to the reference speech sample. Formally, the prosody reference encoder can be represented as a function h that encodes speech representation s into prosody embedding e as $e = h(s)$. We can either use the output of posterior encoder or the extracted linear spectrogram as speech representation s. In our experiments, we find both to have similar performances.

In practice, the reference encoder h is parameterized by a variational encoder module that consists of five convolutional layers of 512 channel size and a kernel size of 3, and one bi-directional LSTM layer of channel size 512. The cell states of LSTM layer is then further processed by two fully connected layers that output the parameterized diagonal Gaussian

distribution, that is regularized using KL-Divergence with a standard Gaussian $\mathcal{N}(0, I)$. The variational Bayesian formulation has two advantages. First, this formulation allows interpolable embedding space, which is conducive for the sampling of prosody. Secondly, carefully tuned KLD regularizes the prosody embedding to reduce speaker and language information contained within the embedding as mentioned in Section 3.4.1, which is essential for cross-speaker and cross-lingual prosody transfer.

We experimented with various ways of conditioning the model on extracted prosody embedding and found that conditioning in the text encoder module (as in Figure 6.1) produces the best result. Intuitively, conditioning in the text encoder allows a joint modelling $P(c, e)$ of the text embedding c and the prosody embedding e, which makes it possible to model long-term dependencies between text sequence and prosody embedding. We denote this system as Variational Inference for Prosody Transfer (VIPT).

6.3.2 Noise Modelling

The prosody encoder, designed to encode the prosody of the reference audio, also encodes other artefacts, such as background noise and distant speech not annotated in the text. In our empirical study, we found that the presence of these artefacts severely degrades the quality of speech synthesis. We propose two approaches to tackle this issue.

In the first approach, we introduce an explicit noise modelling method (see Figure 6.2) to our system, which enables us to disentangle the prosodic information from the noise information. As a result, clean speech audio can be generated even when noisy reference audio is provided. At training time, we use an external denoiser (Isik et al., 2020) to split reference audio into denoised waveform and noise residual waveform. We feed the spectrograms extracted from the two audio streams into separate reference encoders, resulting in two disentangled embeddings: a prosody embedding from denoised audio and a noise embedding from the noise residual. Finally, we concatenate the prosody and noise embeddings to condition the text encoder as described in Section 6.3.1. In this way, a mapping from the noise embedding to noise artefacts contained in the target waveform is learnt. At inference time, the prosody embedding is extracted from a given denoised reference audio containing the desired prosody, while the noise embedding is derived from a separate clean utterance. We denote this system as Variational Inference for Prosody Transfer with Noise Modelling (VIPT-NM).

In the second approach, we use the base VIPT architecture, but input the denoised reference audio during inference time. This approach is able to reduce the noise level in the synthesised speech, but may also introduce distortions due to unseen denoised reference audio input that are out of training data distribution. To remedy this, we add clean data as a proxy for denoised audio to our dataset and train our model with both noisy data and clean data. In this way, the out-of-distribution condition of the denoised reference audio is alleviated, which, in turn, improves the synthesis quality.

6.3.3 Training Setup

We follow the training setup of original VITS (Kim et al., 2021), where we include the use of short-term Fourier transform (STFT) discriminator as in BigVGAN (Lee et al., 2023). The final loss can be expressed as follows:

$$\mathcal{L} = \mathcal{L}_{VITS} + \alpha_1 \mathcal{L}_{ProsodyKLD} + \alpha_2 \mathcal{L}_{NoiseKLD} \tag{6.1}$$

where $\mathcal{L}_{VITS}$ represents the VITS loss terms except with the adversarial loss changed to the BigVGAN's formulation. $\mathcal{L}_{ProsodyKLD}$ and $\mathcal{L}_{NoiseKLD}$ are KLD losses for prosody and noise reference encoders respectively. After hyper-parameter search of 12 runs, we found the best KL-Divergence loss coefficients α_1 and α_2 to be both 0.001. We use mixed precision training on eight NVIDIA V100 GPUs. The batch size is set to 30 per GPU and the models are trained up to 700k steps. The generative part of the VIPT-transfer model has a total of 90 million parameters and discriminators have 47 million parameters.

6.4 Evaluations

We used an internal multi-speaker multilingual dataset mined from existing in-house multimedia source data that contains expressive speech recorded in varying acoustic conditions. The dataset comprises 118 hours of speech recordings from 127 speakers in five different locales; namely US English, Castilian Spanish, French, German and Italian. Speaker age groups range from children to elderly. We split data into training, development, and test sets using a 85:5:10 ratio. For the evaluation of cross-lingual prosody transfer, we ran MUSHRA (Recommendation, 2015) tests on a held-out subset of 100 US English utterances with expressive human dubbing in Castilian Spanish. For the subjective evaluation, for the sake of brevity, we focus on prosody transfer from US English to Castilian Spanish as a representative of other language combinations. Our proposed method also works for other language pairs, and we briefly discuss objective metrics evaluation for the other language pairs in Section 6.4.3. In order to provide testers with a precise context for prosody assessment, we presented the audio samples overlaid on the corresponding videos. We evaluated four systems:

- *VIPT-Centroid* - We aim to pick a baseline model that generates high quality speech, but does not have prosody transfer capability. We introduce VIPT-Centroid, which is the same as VIPT model, but uses the centroid of prosody embeddings calculated across denoised reference samples for the target speaker. VIPT-Centroid is a stronger baseline than other VITS-based external models such as YourTTS (Casanova et al., 2022), because those models, without a reference encoder, tend to internalize the noise into the model parameters and frequently generate noisy speech with higher rate of mispronunciations. To illustrate this, we measured Signal-to-Noise Ratio (SNR) of VIPT-Centroid and YourTTS outputs by using the same denoiser as used in Section 6.3.2. VIPT-Centroid has a SNR of 45.2 dB, which is significantly higher than YourTTS's SNR ratio of 34.8 dB.

- *VIPT-Transfer* - As above but with the prosody embedding extracted from the denoised audio of a source English speaker.
- *VIPT-NM-Transfer* - As above but with explicit noise modelling applied.
- *Recording* - Professional human Spanish dubbing.

In the MUSHRA test, 25 native Spanish speakers were presented with the video samples in a random order side-by-side, and were asked to *"Rate the vocal performance in the Spanish video dubbing samples with respect to the English reference video"*. Each test case was scored by all 25 testers independently.

6.4.1 Perceptual Metrics

Figure 7.3 shows that while VIPT-Transfer on average achieved lower MUSHRA scores than the baseline VIPT-Centroid, VIPT-NM-Transfer was significantly better. On closer inspection, we observed that the VIPT-Transfer system scored lowest for utterances with particularly noisy English reference audios, thereby dragging down the mean score despite of its capability to perform prosody transfer. VIPT-NM-Transfer was more robust to the negative impact of the noise in reference audio and resulted in better matching prosody than the baseline VIPT-Centroid, thereby achieved a statistically significant MUSHRA score increase and closed the gap to human dubbing by 11.2%.

Additionally, we evaluated the effects of adding clean data to improve the synthesis quality when using denoised reference audio. For model training, we added 480 hours of internal clean speech data that consists of 183 additional speakers within the same 5 locales as the original data. Similarly as before, on the cross-lingual prosody transfer from English to Spanish, we used denoised English reference audio to condition the prosody encoder for synthesising Spanish speech with the desired prosody. For the perceptual metric evaluation, we conducted a MUSHRA test with the same setup as above. Figure 6.4 shows the MUSHRA scores of VIPT-Transfer compared to VIPT-Centroid. The improved score of VIPT-Transfer shows that adding clean data allowed us to effectively use a denoised reference audio for performing cross-lingual prosody transfer without a significant compromise in terms of the stability.

6.4.2 Analysis of Prosody Embedding Space

Cross-lingual prosody transfer should only transfer the prosody but not language specific accents to the target language. This requires the prosody embedding space to be disentangled from language categories. In order to verify this and to further understand the learnt VAE reference encoder embedding space, we used t-SNE to reduce dimensions of the embedding space to $\mathcal{R}^2$ and plotted randomly sampled embedding using a colored scatter plot. The embedding was taken from the output mean predicted from the reference encoder for the corresponding reference sample.

Figure 6.5 depicts t-SNE plots of randomly sampled utterances' prosody embedding from five different locales in our dataset with 600 utterances per locale. It can be observed that there is no significant locale clustering, which indicates that the learnt reference embedding space was locale/language independent. This local-independent prosody distribution is essential for performing cross-lingual prosody transfer.

6.4.3 Objective Metrics For Other Language Pairs

In this section, we discuss objective metrics evaluation for language pairs other than US English and Castilian Spanish. As a metric of measure for prosody transfer, we focus on F0 statistics, including Mean Squared Error (MSE) and Pearson correlation between synthesised speech and the corresponding Spanish human dubs. In Table 6.1 F0 objective metrics are computed for an external baseline YourTTS (Casanova et al., 2022), and VIPT-Transfer with different language pairs on the same test set as used for the MUSHRA evaluation. It can be seen that VIPT-Transfer outperforms YourTTS in terms of F0 metrics for any included language pair, which indicates that our proposed method works for more than one language pair. The VIPT-Transfer-En-To-Es system gives the best scores for both metrics, which we hypothesise is due to higher proportion of English utterances in our training data.

Table 6.1: F0 Metrics comparing an external baseline YourTTS (Casanova et al., 2022) and our VIPT-Transfer model with prosody transfer for different language pairs. Mean Squared Error (MSE) and correlation coefficient are computed against corresponding human Spanish recordings.

System	MSE ↓	Correlation ↑
YourTTS	8367.7	0.30
VIPT-Transfer-En-To-Es	6970.0	0.40
VIPT-Transfer-It-To-Es	7639.2	0.35
VIPT-Transfer-Fr-To-Es	7724.7	0.33
VIPT-Transfer-De-To-Es	7693.4	0.34

6.5 Conclusions

We presented a novel solution that learns cross-lingual prosody transfer from non-parallel noisy speech data. We showed that our proposed solution can generate dubbed speech with context-matching prosody. We further demonstrated two approaches to address challenges posed by noise in multimedia data. First, we introduced a novel noise modelling module that disentangles noise from prosody, where a denoised signal extracted from reference audio is utilized. Second, we augment noisy data with clean training data to improve the capability of the model to map denoised reference audio to clean speech. Through subjective and objective evaluations, we showed that our system outperforms a strong baseline in the task of speech generation for automatic dubbing.

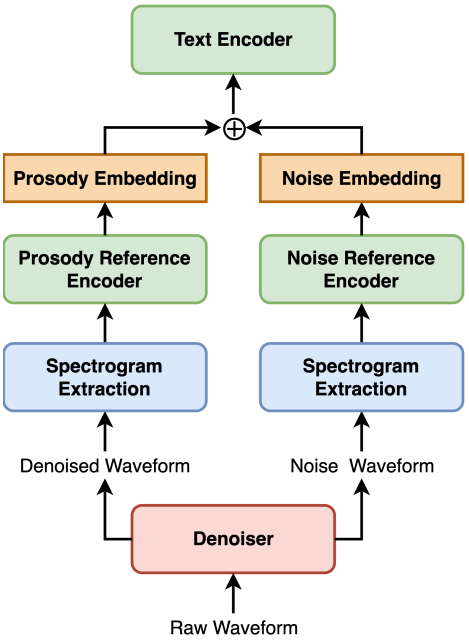


Figure 6.2: Explicit noise modelling method utilizing both prosody and noise reference encoders.

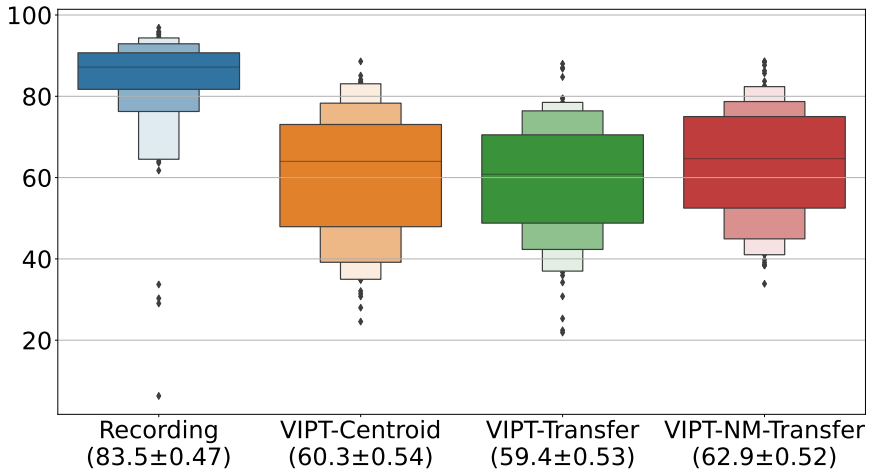


Figure 6.3: Subjective listeners ratings from the machine dubbing MUSHRA test for VIPT and VIPT-NM. Values under labels represent mean scores and their respective standard errors.

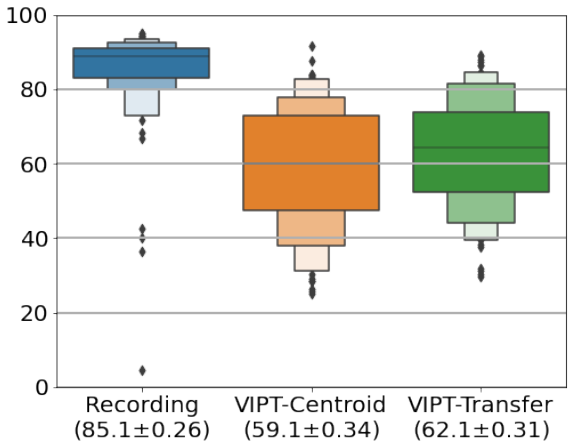


Figure 6.4: Subjective listeners ratings from the cross-lingual prosody transfer MUSHRA test for VIPT-Transfer with additional clean training data. Values under labels represent mean scores and their respective standard errors.

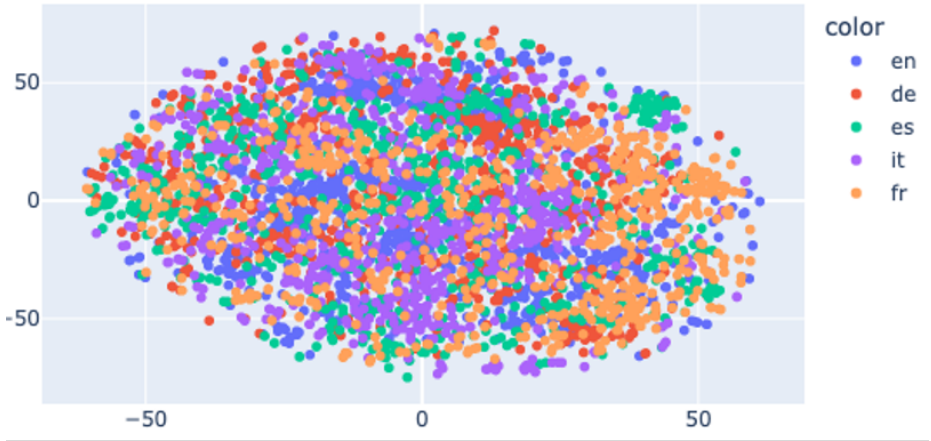


Figure 6.5: T-SNE plot of VAE reference encoder embedding space coloured by languages.

Chapter 7

Expressive Machine Dubbing Through Phrase-level Cross-lingual Prosody Transfer

7.1 Overview

Speech generation for machine dubbing adds complexity to conventional Text-To-Speech solutions as the generated output is required to match the expressiveness, emotion and speaking rate of the source content. Capturing and transferring details and variations in prosody is a challenge. In this chapter, we introduce phrase-level cross-lingual prosody transfer for expressive multi-lingual machine dubbing. The proposed phrase-level prosody transfer delivers a significant 6.2% MUSHRA score increase over a baseline with utterance-level global prosody transfer from Chapter 6, thereby closing the gap between the baseline and expressive human dubbing by 23.2%, while preserving intelligibility of the synthesised speech.

7.2 Introduction

As defined in the previous chapters, prosody transfer is the ability to transfer speaking style variations and vocal performances disentangled from the spoken content and speaker identity (Skerry-Ryan et al., 2018b; Wang et al., 2018; Luong et al., 2017; Hsu et al., 2019; Zhang et al., 2019b). Many of recent proposed prosody transfer methods utilize global-level prosody transfer. A single embedding per utterance is used to capture prosody and to condition the models to generate speech with the target prosody. These global embeddings are either explicitly learned from ground truth labels such as emotions (Luong et al., 2017; Guo et al., 2023; Rattcliffe et al., 2022b) or implicitly learned from a reference audio signal using a reference prosody encoder (Skerry-Ryan et al., 2018b; Wang et al., 2018; Zhang et al., 2019b), or a combination of both (Hsu et al., 2019).

In this chapter, we keep our focus from Chapter 6 on prosody transfer for cross-lingual machine dubbing. To our best knowledge, VIPT (Variational Inference for Prosody Transfer) described in Chapter 6, is the only known work tackling cross-lingual prosody transfer for machine dubbing. VIPT uses a global reference encoder to capture prosody. One limitation in using a global reference embedding is that only utterance-level prosody variations are captured, while detailed local prosody variations cannot be properly encoded. This potentially impacts the transfer of prosody for generating long-form utterances. Transferring local prosody variations such as syntactic phrasing, topic emphasis and marked tonicity is important for expressive machine dubbing (Torresquintero et al., 2021). We tackle this drawback by exploring more fine-grained cross-lingual prosody transfer.

Intra-lingual fine-grained prosody control has been explored (Ren et al., 2020; Lee & Kim, 2019; Babiański et al., 2023; Sun et al., 2020) by training predictors of prosody components at phoneme or at word level. However, prosody transfer across different utterances at word level suffers due to word mismatch. This applies as well to cross-lingual prosody transfer for machine dubbing, as it is unlikely to have one-to-one alignment between words in the original and translated text. Several recent works (Virkar et al., 2021; Effendi et al., 2022a) have proposed machine translation techniques for machine dubbing, allowing to generate monotonic alignments between translated texts at the level of a prosodic phrase, where a prosodic phrase is defined as a continuous segment of speech separated by silence regions. Therefore, we explore phrase-level cross-lingual prosody transfer for machine dubbing.

Prosody delivery varies across languages, however the prosody of speech expressing the same emotions is correlated in related languages, as discussed in Section 4.6 in Brannon et al. (2021). We explore these cross-lingual correlations for the purpose of prosody transfer. Our study is limited to European languages comprising English, German, French, Italian and Spanish, and focused on English-Spanish prosody transfer as a common dubbing language pair. We anticipate that more distant language pairs such as English-Japanese exhibit less correlated prosody features.

Our solution follows VIPT from Chapter 6 in combining a VAE (Variational Auto-Encoder) prosody encoder with VITS and trains on multimedia data without mining of parallel utterances with matching text across different locales. We propose to capture and to transfer phrase-level variations of prosody in a cross-lingual setting. To achieve that, we have devised a new phrase-level reference encoder that learns to condition the phrases of the input text with prosody embeddings extracted from corresponding parts of the reference speech waveform. We have also introduced a novel regularization applied on prosody embeddings based on phrase length, to reduce content leakage from short phrases. We discuss the details in Section 7.3.

We evaluate our proposed method with both MUSHRA (Recommendation, 2015) subjective perceptual test and objective metrics including Word Error Rate (WER) and conditional Fréchet DeepSpeech Distance (cFDSD) (Bińkowski et al., 2020). We compare our method against VIPT from Chapter 6, a strong baseline for cross-lingual performance transfer. Both subjective and objective metrics suggest that our method improves expressiveness without compromising on intelligibility. We also show the importance of phrase-level conditioning in

training, by comparing against a VIPT variant trained with global-level conditioning, but transferring prosody at phrase-level during inference. We demonstrate a significant 6.2% MUSHRA score increase over VIPT, which closes the gap between machine dubbing and expressive human dubbing by 23.2%. To summarize, our contributions are:

- We present a new method capable of cross-lingual phrase-level prosody transfer for expressive multi-lingual machine dubbing. Robust and more fine-grained transfer compared to global-level prosody transfer improves the quality.
- We propose a length-based regularization method for fine-grained prosody representations.

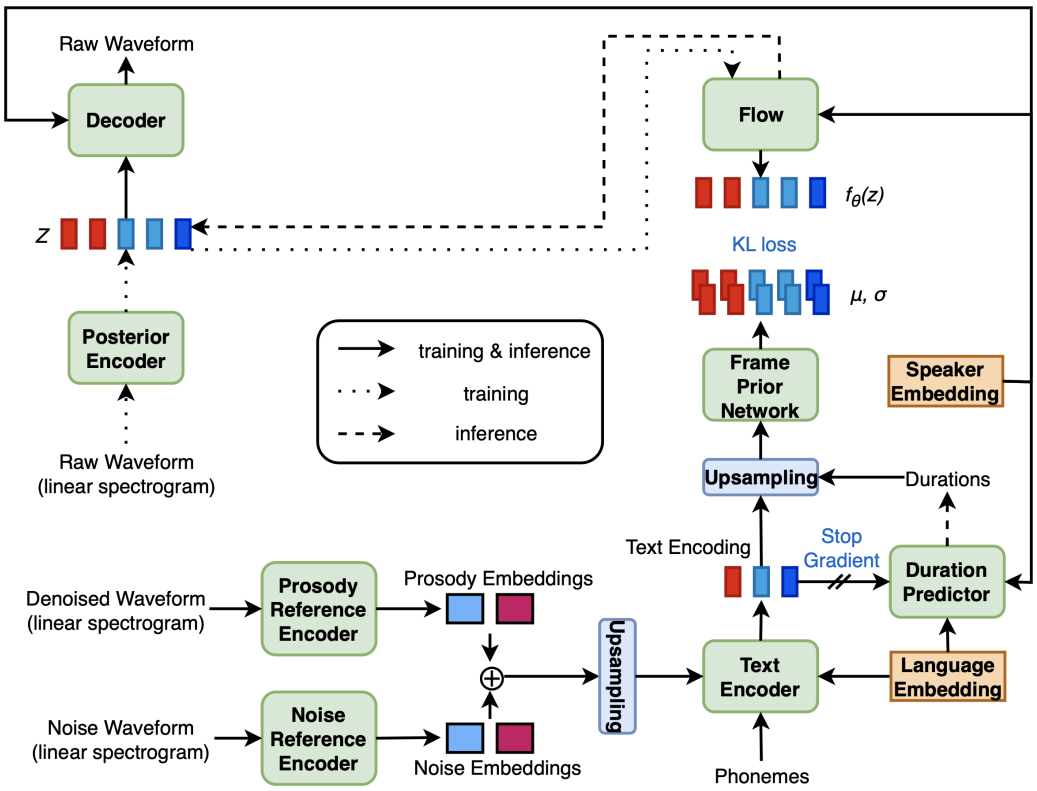


Figure 7.1: Proposed system architecture.

7.3 Method

This work extends the VIPT architecture from Chapter 6 by enabling modelling and cross-lingual transfer of prosody at the phrase level. Figure 7.1 provides an overview of the proposed method. This section describes extensions to the VIPT architecture from Chapter 6 that enable the modelling and cross-lingual prosody transfer at the phrase level.

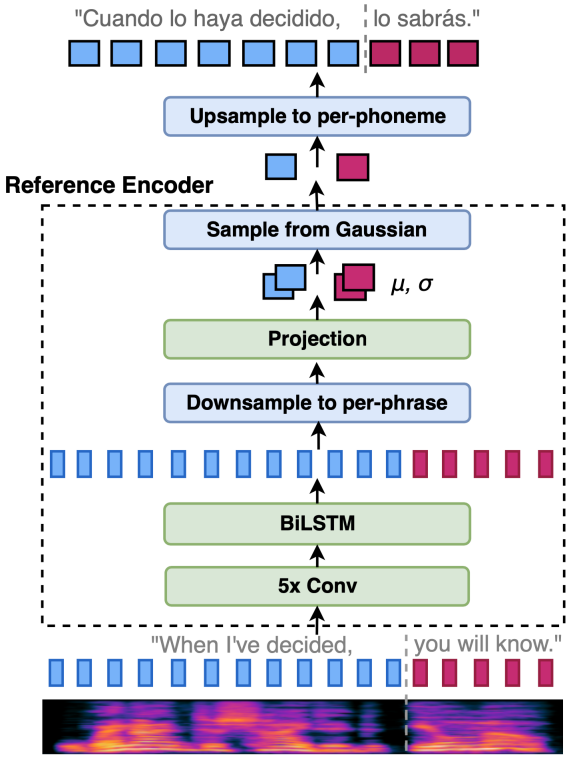


Figure 7.2: Phrase-level reference encoder.

The global-level reference encoder in VIPT may not sufficiently transfer the prosody variations present in expressive multimedia speech, especially when a dialogue line consists of several short phrases. Capturing prosody variations consequently requires more fine-grained representations. Word-level prosody representations are challenging to transfer in a cross-lingual setting due to lack of monotonic word correspondence between translated texts. Instead, we propose prosodic phrases as a level of granularity for cross-lingual prosody transfer. We show experimentally that prosodic phrases are able to capture local variations in prosody which can be robustly transferred between speech in different languages. At the same time, prosodic phrases can be automatically aligned across translated texts using recently developed prosodic alignment techniques for machine dubbing (Virkar et al., 2021; Effendi et al., 2022a).

7.3.1 Phrase-level reference encoder

We adopt the definition of prosodic phrases from (Virkar et al., 2021; Effendi et al., 2022a) as continuous speech segments separated by silences. The silences are extracted by force aligning reference audio and text using an external aligner, such as the Gaussian Mixture Model (GMM) based Kaldi Speech Recognition Toolkit (Povey et al., 2011) used in our experiments. We treat the silence regions as part of preceding speech phrases. Each speech

phrase is encoded into a single prosody embedding using a reference encoder described below. See Figure 7.2 for an illustration of this approach.

Our proposed phrase-level reference encoder architecture extracts frame-level embeddings from a linear spectrogram and downsamples the frame-level embeddings to the phrase-level. We base our reference encoder architecture on that from VIPT, but with changes to keep one-to-one frame-embedding correspondence before downsampling to the phrase-level. Namely, our architecture consists of five convolutional layers with a channel size of 512, a kernel size of three and a stride of one, followed by one bi-directional LSTM layer with channel size of 512. The frame-level outputs of the bi-directional LSTM are then downsampled by selecting the middle embedding per phrase. We experimented with other forms of downsampling (e.g. mean of frames per phrase) but did not observe significant differences. The phrase-level embeddings are then further processed by a fully connected layer that outputs a parameterization of a 32-dimensional diagonal Gaussian distribution, which is regularized using Kullback-Leibler Divergence (KLD) with a standard Gaussian $\mathcal{N}(0, I)$. The final phrase embeddings are sampled from this Gaussian.

7.3.2 Length-based regularization

We propose a length-based regularization of phrase-level prosody embeddings to reduce content leakage when transplanting prosody embeddings across languages. As we have observed content leakage for short phrases, we added a length-based regularization as following:

$$\mathcal{L}_{KLD_\beta} = \frac{1}{K} \sum_{k \in [1,K]} e^{-\beta L_k} KLD(h_k, \mathcal{N}(0, I)) \tag{7.1}$$

where K is the total number of phrases in one utterance, L_k is length of phrase k defined as the number of phonemes, β is a constant hyper-parameter controlling how much L_k affects the scaling factor $e^{-\beta L_k}$, h_k is the prosody embedding distribution of the k^{th} phrase. This formulation applies stronger regularization to the embeddings of short phrases, thus preventing them from carrying content information that should only be obtained from the text prior. We show experimentally that the proposed regularization improves the intelligibility of synthesised speech for short phrases in Table 7.2.

7.3.3 Noise modelling at phrase-level

We apply the noise modelling approach from VIPT in the context of the phrase-level reference encoder. Specifically, the reference audio is passed to a denoising component (Isik et al., 2020) to extract denoised and noise streams. The two streams are then used as inputs to two separate phrase-level reference encoders with the architecture described in subsection 7.3.1. The reference encoders output phrase-level denoised prosody and noise embeddings that are concatenated, upsampled to the phoneme level and used as conditioning in the text encoder.

During inference, we extract a clean noise embedding from a static clean audio similarly to VIPT. However, we make sure to use a clean audio, which contains exactly one phrase, so

that it is feasible to upsample this single clean noise embedding to match the number of the per-phrase prosody embeddings extracted from the denoised reference audio.

7.3.4 Alignment of phrase-level audio reference embeddings to target text phonemes

We aim to transfer prosody from phrases of reference speech to corresponding phrases in the translated target text to be synthesised. More precisely, we concatenate phrase embeddings extracted from the reference audio with encoded phonemes corresponding to a given phrase in the target text. To achieve this, we need a mapping between reference audio phrases and target text phonemes. See the illustration in Figure 7.2.

During training, the reference audio and the text to be synthesised correspond to each other. This allows us to force align the audio and the text phonemes to compute frame-phoneme correspondences. During inference, when performing cross-lingual prosody transfer, the reference audio contains speech in a language different from the translated text to be synthesised. Therefore, in such case we cannot align the audio and the text as in training. Instead, we need to insert phrase breaks into the translated text to obtain a monotonic one-to-one alignment between the phrases in the audio and the text. Such alignments can be automatically generated using recently developed prosodic alignment techniques for machine dubbing (Virkar et al., 2021; Effendi et al., 2022a). However, the focus of this work is on evaluating the quality of prosody transfer, and thus we assume the prosodic alignment is given.

7.4 Experiments

7.4.1 Training setup

Our training setup largely follows that of VIPT described in Section 6.3.3 except updating the KLD regularization for prosody $\mathcal{L}_{ProsodyKLD}$ and noise $\mathcal{L}_{NoiseKLD}$ reference encoders with the phrase-level formulation and the length-based scaling coefficients β described in Section 7.3.2. The final loss can be formulated as:

$$\mathcal{L} = \mathcal{L}_{VITS} + \alpha_1 \mathcal{L}_{ProsodyKLD_{\beta_1}} + \alpha_2 \mathcal{L}_{NoiseKLD_{\beta_2}} \tag{7.2}$$

where $\mathcal{L}_{VITS}$ represents the VITS loss terms with replaced adversarial components as in BigVGAN (Lee et al., 2023). We performed nine runs of hyperparameter search and set the KLD loss coefficient α and the length-based KLD loss scaling coefficient β for both the prosody and the noise reference encoder as $\alpha_1 = \alpha_2 = 0.04$ (other tested values: 0.02 and 0.08) and $\beta_1 = \beta_2 = 0.08$ (other tested values: 0.02 and 0.04) respectively. We trained using mixed precision on 8 NVIDIA V100 GPUs, with a batch size of 30 per GPU, and used AdamW optimizer (Loshchilov & Hutter, 2019). We trained the model for 600 epochs. The generative part of our proposed model and discriminators have 100 million and 47 million parameters respectively.

7.4.2 Data

We used an internal multimedia dataset from which we extract multi-speaker multi-lingual dialogues resulting in 598 hours of speech from 134 female and 162 male speakers in 5 different locales; namely US English, Castilian Spanish, French, German and Italian. Speaker age groups range from children to elderly.

The speech data is resampled to 24 kHz and normalized in terms of loudness. Silences longer than 2 seconds are trimmed. We split the dialogues into separate phrases based on silences of at least 50 milliseconds.

7.4.3 Evaluated systems

We evaluated the proposed method against human Spanish dubs and two baseline models. We denote our method as Variational Inference for Prosody Transfer with Noise Modelling and Phrase-level Variational Auto-Encoder (VIPT-NM-PVAE). VIPT-NM-GVAE is a strong baseline for cross-lingual performance transfer with a global-level reference encoder (corresponding to VIPT-NM-Transfer model from Chapter 6). Additionally, we introduce a second baseline named VIPT-NM-GVAE-PP, which uses the same model architecture as VIPT-NM-GVAE during training, while at inference time, it computes prosody embeddings per phrase (PP). Namely, during inference, the VIPT-NM-GVAE-PP model passes parts of the source audio corresponding to each of the K phrases separately through the global-level reference encoder to extract the K phrase-level embeddings. We include this baseline to evaluate the importance of training phrase-level embeddings.

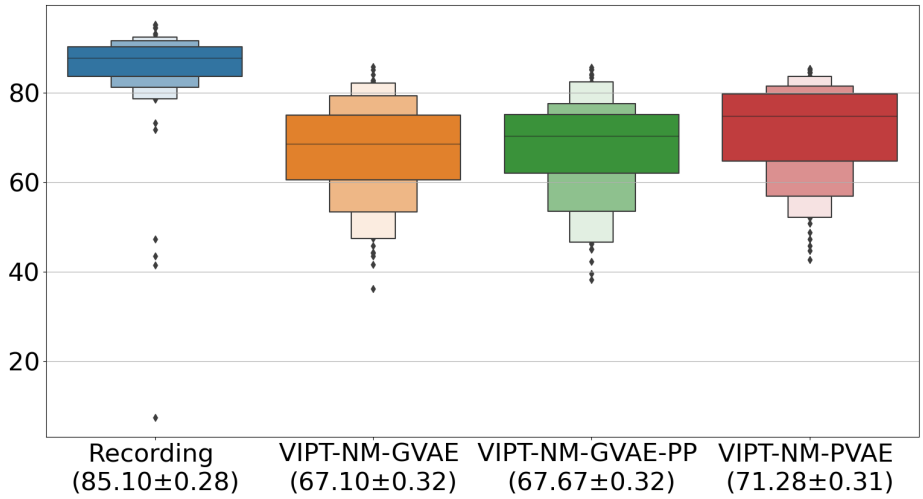


Figure 7.3: Subjective listeners ratings from the machine dubbing MUSHRA test. Values under labels represent mean scores and their respective standard errors.

Table 7.1: Subjective evaluation MUSHRA mean scores reported separately for single-phrase and multi-phrase utterances.

System	MUSHRA ↑	
	single-phrase	multi-phrase
VIPT-NM-GVAE	69.58 ± 0.40	63.21 ± 0.52
VIPT-NM-GVAE-PP	69.81 ± 0.39	64.32 ± 0.53
VIPT-NM-PVAE	$\mathbf{74.31 \pm 0.36}$	$\mathbf{66.56 \pm 0.53}$
Recording	83.75 ± 0.38	87.22 ± 0.40

7.4.4 Subjective Evaluation

For the evaluation of cross-lingual prosody transfer, we performed a MUSHRA test on a held-out subset of 100 parallel utterances between US English and Castilian Spanish. To provide testers context for the assessment of prosody match, all audio samples were overlaid on the corresponding videos. 25 bi-lingual test subjects native in Castilian Spanish and fluent in English were presented with the video samples in a random order side-by-side. The test subjects were tasked to *"Rate the vocal performance in the Spanish video dubbing samples with respect to the English reference video"*. Each test case was scored by all 25 testers independently.

Evaluation results are summarized in Figure 7.3 and show that VIPT-NM-PVAE achieved a statistically significant 6.2% MUSHRA score increase over VIPT-NM-GVAE baseline system, which closes the gap to human dubbing by 23.2%. Inspection of evaluators ratings suggests that the improvement in VIPT-NM-PVAE results from increased expressiveness of generated speech and more accurate prosody transfer. There is significant difference in MUSHRA scores between VIPT-NM-PVAE and both baseline models for multi-phrase utterances, and for single-phrase utterances (Table 7.1). We hypothesize that training with the proposed phrase-level reference encoder may lead to increased sensitivity to the prosody embedding.

Table 7.2: Objective metrics: word error rate (WER) and conditional Fréchet DeepSpeech Distance (cFDSD) (Bińkowski et al., 2020).

System	cFDSD ↓	WER ↓	
		all	shortest 25%
VIPT-NM-GVAE	0.297	0.098	0.169
VIPT-NM-GVAE-PP	0.288	0.094	0.155
VIPT-NM-PVAE	**0.224**	0.101	0.161
w/o length-based reg.	0.241	0.106	0.229

7.4.5 Objective Metrics

To quantify stability of tested systems and intelligibility of synthesised speech we conducted Word Error Rate (WER) analysis. The results are reported in Table 7.2 for a held-out test set of 1200 parallel utterances. First, all generated audio files were transcribed with a Whisper Large (Radford et al., 2022) ASR model. Then, WER scores were computed between sentence texts and corresponding transcriptions. We have observed no significant stability issues with the VIPT-NM-PVAE model, which backs up our conclusion that phrase-level modelling allows for more expressive and accurate cross-lingual prosody transfer without compromising intelligibility.

For all tested systems we also computed the conditional Fréchet DeepSpeech Distance (cFDSD) (Bińkowski et al., 2020), an objective metric measuring the quality of synthesised audio samples based on their distance to a reference set. We closely follow Bińkowski et al. (2020) in our implementation of the cFDSD metric, only differing in using XLSR-53 Large (Conneau et al., 2022) as a backbone network, which was trained on multi-lingual speech data. All tested systems are compared to human Spanish dubs. We observe that VIPT-NM-PVAE has a significantly lower distance to the human dubs (Table 7.2) compared to all other models. This result is inline with the MUSHRA subjective evaluation scores.

Finally, as an ablation study, we trained a VIPT-NM-PVAE model without the length-based regularization described in Section 7.3.2. This resulted in a significant WER increase for short utterances (Table 7.2), while at the same time cFDSD distance to recordings also increased. We conclude that applying regularization dependent on phrase lengths is crucial to find a good balance between expressivity and stability of our system.

7.5 Conclusions

We have presented a novel solution that enables phrase-level cross-lingual, cross-speaker prosody transfer for expressive machine dubbing. The proposed method can learn to model prosody information at phrase-level, and transfer the phrase prosody embeddings from a source to a target language for translated text. In subjective evaluations, our system outperforms a strong baseline that transfers prosody at global-level. In future work, we plan to extend our evaluation to include wider range of languages and further close the gap between synthesised speech and expressive human dialogues by exploring duration modelling, hierarchical prosody modelling and the usage of parallel data.

Appendix A

Appendix for Chapter 4

A.1 Proof of the Matrix Variate Normal Parameterization

In this section of the appendix, we formally explain the connections between the k-tied Normal distribution and the matrix variate Gaussian distribution (Gupta & Nagar, 2018), referred to as $\mathcal{MN}$.

Consider positive definite matrices $\mathbf{Q} \in \mathbb{R}^{r\times r}$ and $\mathbf{P} \in \mathbb{R}^{c\times c}$ and some arbitrary matrix $\mathbf{M} \in \mathbb{R}^{r\times c}$. We have by definition that $\mathbf{W} \in \mathbb{R}^{r\times c} \sim \mathcal{MN}(\mathbf{M}, \mathbf{Q}, \mathbf{P})$ if and only if $\text{vec}(\mathbf{W}) \sim \mathcal{N}(\text{vec}(\mathbf{M}), \mathbf{P} \otimes \mathbf{Q})$, where $\text{vec}(\cdot)$ stacks the columns of a matrix and $\otimes$ is the Kronecker product

The $\mathcal{MN}$ has already been used for variational inference by Louizos & Welling (2016) and Sun et al. (2017). In particular, Louizos & Welling (2016) consider the case where both $\mathbf{P}$ and $\mathbf{Q}$ are restricted to be diagonal matrices. In that case, the resulting distribution corresponds to our k-tied Normal distribution with $k = 1$ since

$$\mathbf{P} \otimes \mathbf{Q} = \text{diag}(\mathbf{p}) \otimes \text{diag}(\mathbf{q}) = \text{diag}(\text{vec}(\mathbf{q}\mathbf{p}^\top)).$$

Importantly, we prove below that, in the case where $k \geq 2$, the k-tied Normal distribution cannot be represented as a matrix variate Gaussian distribution.

Lemma (Rank-2 matrix and Kronecker product)**.** *Let* $\mathbf{B}$ *be a rank-*2 *matrix in* $\mathbb{R}_+^{r\times c}$*. There do not exist matrices* $\mathbf{Q} \in \mathbb{R}^{r\times r}$ *and* $\mathbf{P} \in \mathbb{R}^{c\times c}$ *such that*

$$\textit{diag}(\textit{vec}(\mathbf{B})) = \mathbf{P} \otimes \mathbf{Q}.$$

Proof. Let us introduce the shorthand $\mathbf{D} = \text{diag}(\text{vec}(\mathbf{B}))$. By construction, $\mathbf{D}$ is diagonal and has its diagonal terms strictly positive (it is assumed that $\mathbf{B} \in \mathbb{R}_+^{r\times c}$, i.e., $b_{ij} > 0$ for all i, j).

We proceed by contradiction. Assume there exist $\mathbf{Q} \in \mathbb{R}^{r\times r}$ and $\mathbf{P} \in \mathbb{R}^{c\times c}$ such that $\mathbf{D} = \mathbf{P} \otimes \mathbf{Q}$.

This implies that all diagonal blocks of $\mathbf{P} \otimes \mathbf{Q}$ are themselves diagonal with strictly positive diagonal terms. Thus, $p_{jj}\mathbf{Q}$ is diagonal for all $j \in \{1, \ldots, c\}$, which implies in

turn that $\mathbf{Q}$ is diagonal, with non-zero diagonal terms and $p_{jj} \neq 0$. Moreover, since the off-diagonal blocks $p_{ij}\mathbf{Q}$ for $i \neq j$ must be zero and $\mathbf{Q} \neq \mathbf{0}$, we have $p_{ij} = 0$ and $\mathbf{P}$ is also diagonal.

To summarize, if there exist $\mathbf{Q} \in \mathbb{R}^{r\times r}$ and $\mathbf{P} \in \mathbb{R}^{c\times c}$ such that $\mathbf{D} = \mathbf{P} \otimes \mathbf{Q}$, then it holds that $\mathbf{D} = \text{diag}(\mathbf{p}) \otimes \text{diag}(\mathbf{q})$ with $\mathbf{p} \in \mathbb{R}^c$ and $\mathbf{q} \in \mathbb{R}^r$. This last equality can be rewritten as $b_{ij} = p_j q_i$ for all $i \in \{1, \dots, r\}$ and $j \in \{1, \dots, c\}$, or equivalently

$$\mathbf{B} = \mathbf{q}\mathbf{p}^\top.$$

This leads to a contradiction since $\mathbf{q}\mathbf{p}^\top$ has rank one while $\mathbf{B}$ is assumed to have rank two. □

Figure A.1 provides an illustration of the difference between the k-tied Normal and the $\mathcal{MN}$ distribution.

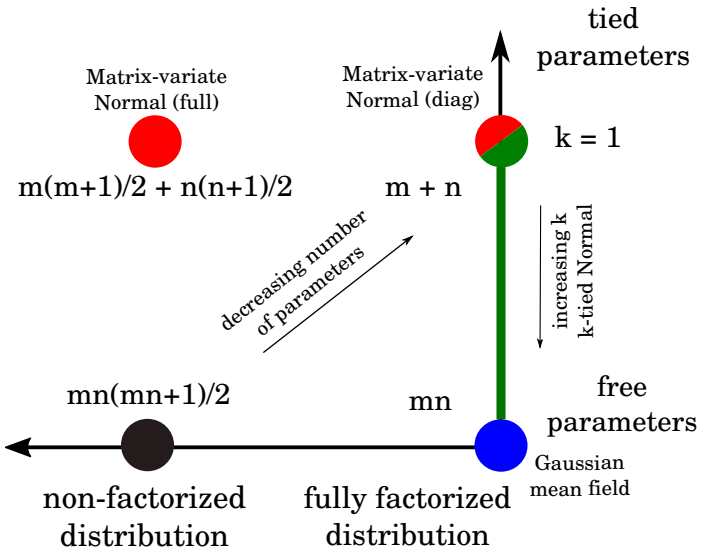


Figure A.1: Illustration of the difference in modeling of the posterior covariance by the k-tied Normal distribution (green), the $\mathcal{MN}$ distribution (red), the Gaussian mean field (blue) and the dense Gaussian covariance (black) for a layer of size $m \times n$. The k-tied Normal with $k = 1$ is equivalent to $\mathcal{MN}$ with diagonal row and column covariance matrices (half-red, half-green circle). Our experiments show that the $k = 1$ fails to capture the performance of the mean field. On the other hand, while the full/non-diagonal $\mathcal{MN}$ increases the expressiveness of the posterior, it also increases the number of parameters. In contrast, the k-tied Normal distribution with $k \geq 2$ not only decreases the number of parameters, but also matches the predictive performance of the mean field.

A.2 He-scaled Normal Prior

We investigate whether the low-rank structure is specific to the GMFVI neural networks that use a Normal prior with a single scalar scale for all the weights. Instead of using the single

scale parameter, we analyse a setting in which the Normal prior scale is set according to the scaling rules devised for neural network weights initialization Glorot & Bengio (2010b); He et al. (2015a). According to these rules, a per layer scale parameter is set according to the layer shape and activation function used. In particular, we use the scaling rule from He et al. (2015a) for the models with ReLU activations Glorot et al. (2011):

$$p(\mathbf{w}_l) = \mathcal{N}\left(0, \frac{2}{m_l}\right), \tag{A.1}$$

where m_l is the *fan-in* of the m'th layer.[1] However, the scaling rule proposed in He et al. (2015a) does not cover the bias terms, which are initialized at zero. Therefore, for the ResNet-18 on CIFAR-10 which we take under test, we keep the prior for the biases unchanged at $\mathcal{N}(0, I)$. We rerun then the low-rank structure experiments from Section 4.3.3 Figure 4.5, but now with the He-scaled prior. Figure A.2 shows the low-rank structure analysis results for the new prior. While we observe an overall drop in performance, the low-rank structure clearly remains present.

A.3 KL Annealing with Adam

We verify that when using KL annealing with Adam the posterior standard deviation parameters do not converge prematurely, but rather continue being optimized after the KL is at its full contribution. Figure A.3 illustrates this on the example of the ResNet-18 CIFAR-10 model trained the standard GMFVI. Furthermore, for the MLP, CNN and LSTM models, we observed their posterior standard deviations at convergence to have large values compared to the prior standard deviation value (>50% of the prior value), showing that we are modeling substantial uncertainty.

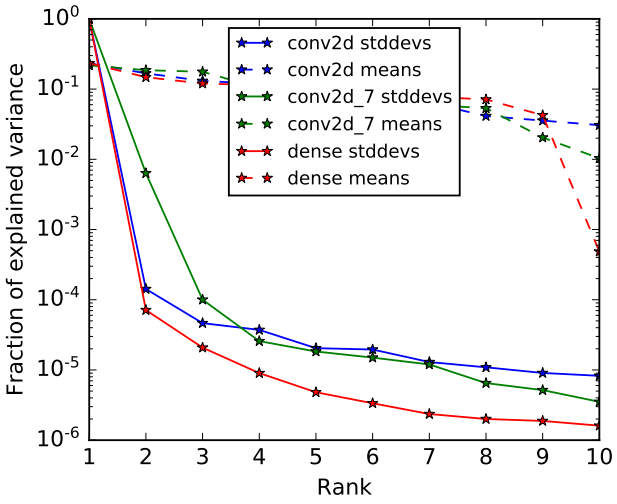
A.4 Experimental Details

In this section we provide additional information on the experimental setup used in Chapter 4. In particular, we describe the details of the models and datasets, the utilized standard Gaussian Mean Field Variational Inference (GMFVI) training procedure, the low-rank structure analysis of the GMFVI trained posteriors and the proposed k-tied Normal posterior training procedure.

A.4.1 Models and datasets

To confirm the validity of our results, we performe the experiments on a range of models and datasets with different data types, architecture types and sizes. Below we describe their details.

[1] For a dense layer the fan-in is the number of input dimensions, for a 2D Convolutional layer with a kernel of size $k \times k$ and d input channels the fan-in is $m_l = k^2 d$.



Method	-ELBO ↓	NLL ↓	Accuracy ↑
Mean-field	$1.379_{\pm 0.0096}$	$0.6384_{\pm 0.0096}$	$79.0_{\pm 0.41}$
1-tied	$5.428_{\pm 0.018}$	$1.485_{\pm 0.0056}$	$57.0_{\pm 0.50}$
2-tied	$1.448_{\pm 0.0097}$	$0.648_{\pm 0.0079}$	$78.8_{\pm 0.41}$
3-tied	$1.411_{\pm 0.0097}$	$0.646_{\pm 0.0079}$	$78.9_{\pm 0.41}$

Figure A.2: The post-training low rank structure is still present in the posterior standard deviation parameters of the ELBO-converged standard GMFVI ResNet-18 CIFAR-10 model when using the He-scaled prior. Approximations to these parameters with ranks higher than 1 result in performance close to that when not using the approximation. We report mean and SEM for predictions made using an ensemble of 100 weights samples. The SEM is measured across the test examples.

MLP MNIST Multilayer perceptron (MLP) model with three dense layers and ReLu activations trained on the MNIST dataset (LeCun & Cortes, 2010). The three layers have sizes of 400, 400 and 10 hidden units. We preprocess the images to be have values in range $[-1, 1]$. We use the last 10,000 examples of the training set as a validation set.

LeNet CNN CIFAR-100 LeNet convolutional neural network (CNN) model (LeCun et al., 1998) with two convolutional layers followed by two dense layers, all interleaved with ReLu activations. The two convolutional layers have 32 and 64 output filters respectively, each produced by kernels of size 3×3. The two dense layers have sizes of 512 and 100 hidden units. We train this network on the CIFAR-100 dataset (Krizhevsky et al., 2009b). We preprocess the images to have values in range $[0, 1]$. We use the last 10,000 examples of the training set as a validation set.

LSTM IMDB Long short-term memory (LSTM) model (Hochreiter & Schmidhuber, 1997) that consists of an embedding and an LSTM cell, followed by a dense layer with a single unit.

The LSTM cell consists of two dense weight matrices, namely the kernel and the recurrent kernel. The embedding and the LSTM cell have both 128-dimensional output space. More precisely, we adopt the publicly available LSTM Keras (Chollet et al., 2015) example[2], except that we set the dropout rate to zero. We train this model on the IMDB text sentiment classification dataset (Maas et al., 2011), in which we use the last 5,000 examples of the training set as a validation set.

ResNet-18 CIFAR-10 ResNet-18 model (He et al., 2016a) trained on the CIFAR-10 dataset (Krizhevsky et al., 2009b). We adopt the ResNet-18 implementation[3] from the Tensorflow Probability (Dillon et al., 2017) repository. We train/evaluate this model on the train/test split of 50,000 and 10,000 images, respectively, from the CIFAR-10 dataset available in Tensorflow Datasets[4].

A.4.2 GMFVI training

We train all the above models using GMFVI. We split the discussion of the details of the GMFVI training procedure into two parts. First, we describe the setup for the MLP, CNN and LSTM models, for which we prepare our own GMFVI implementations. Second, we explain the setup for the GMFVI training of the ResNet-18 model, for which we use the implementation available in the Tensorflow Probability repository as mentioned above.

MLP, CNN and LSTM In the MLP and the CNN models, we approximate the posterior using GMFVI for all the weights (both kernel and bias weights). For the LSTM model, we approximate the posterior using GMFVI only for the kernel weights, while for the bias weights we use a point estimate. For all the three models, we use the standard reparametrization trick estimator (Kingma & Welling, 2013). We initialize the GMFVI posterior means using the standard He initialization (He et al., 2015a) and the GMFVI posterior standard deviations using samples from $\mathcal{N}(0.01, 0.001)$. Furthermore, we use a Normal prior $\mathcal{N}(0, \sigma_p I)$ with a single scalar standard deviation hyper-parameter σ_p for all the layers. We select σ_p for each of the models separately from a set of $\{0.2, 0.3\}$ based on the validation data set performance.

We optimize the variational parameters using an Adam optimizer (Kingma & Ba, 2014). We pick the optimal learning rate for each model from the set of $\{0.0001, 0.0003, 0.001, 0.003\}$ also based on the validation data set performance. We choose the batch size of 1024 for the MLP and CNN models, and the batch size of 128 for the LSTM model. We train all the models until the ELBO convergence.

To implement the MLP and CNN models we use the `tfp.layers` module from the Tensorflow Probability, while to implement the LSTM model we use the `LSTMCellReparameterization`[5]

[2]See: `https://github.com/keras-team/keras/blob/master/examples/imdb_lstm.py`.

[3]See: `https://github.com/tensorflow/probability/blob/master/tensorflow_probability/examples/cifar10_bnn.py`.

[4]See: `https://www.tensorflow.org/datasets/catalog/cifar10`.

[5]See: `https://github.com/google/edward2/blob/master/edward2/tensorflow/layers/recurrent.py`.

class from the Edward2 Layers module Tran et al. (2019).

ResNet-18 The specific details of the GMFVI training of the ResNet-18 model can be found in the previously linked implementation from the Tensorflow Probability repository. Here, we describe the most important and distinctive aspects of this implementation.

The ResNet-18 model approximates the posterior using GMFVI only for the kernel weights, while for the bias weights it uses a point estimate. The model uses the Flipout estimator (Wen et al., 2018) and a constraint on the maximum value of the GMFVI posterior standard deviations of 0.2. The GMFVI posterior means are initialized using samples from $N(0, 0.1)$, while the GMFVI posterior log standard deviations are initialized using samples from $\mathcal{N}(-9.0, 0.1)$. Furthermore, the model uses a Normal prior $\mathcal{N}(0, I)$ for all of its layers.

The variational parameters are trained using the Adam optimizer with a learning rate of 0.0001 and a batch size of 128. The model is trained for 700 epochs. The contribution of the D_{KL} term in the negative Evidence Lower Bound (ELBO) equation is *annealed* linearly from zero to its full contribution over the first 50 epochs (Sønderby et al., 2016).

A.4.3 Low-rank structure analysis

After training the above models using GMFVI, we investigate the low-rank structure in their trained variational posteriors. For the MLP, CNN and LSTM models, we investigate the low-rank structure of their dense layers only. For the ResNet-18 model, we investigate both its dense and convolutional layers.

To investigate the low-rank structure in the GMFVI posterior of a dense layer, we inspect a spectrum of the posterior mean and standard deviation matrices. In particular, for both the posterior mean and standard deviation matrices, we consider the fraction of the variance explained by the top singular values from their SVD decomposition (see Figure 4.3). Furthermore, we explore the impact on predictive performance of approximating the reshaped diagonal matrices with their low-rank approximations using only the components corresponding to the top singular values (see Table 4.2). Note that such low-rank approximations may contain values below zero. This has to be addressed when approximating the matrices of the posterior standard deviations, which can contain only positive values. Therefore, we use a lower bound of zero for the values of the approximations to the posterior standard deviations.

To investigate the low-rank structure in a GMFVI posterior of a convolutional layer, we need to add a few more steps compared to those for a dense layer. In particular, weights of the convolutional layers considered here are 4-dimensional, instead of 2-dimensional as in the dense layer. Therefore, before performing the SVD decomposition, as for the dense layers, we first reshape the 4-dimensional weight tensor from the convolutional layer into a 2-dimensional weight matrix. More precisely, we flatten all dimensions of the weight tensor except for the last dimension (e.g., a weight tensor of shape $[3, 3, 512, 512]$ is reshaped to $[3 \cdot 3 \cdot 512, 512]$). Figure A.4 contains example visualizations of the resulting flattened 2-dimensional matrices[6].

[6]After this specific reshape operation, all the weights corresponding to a single output filter are contained in a single column of the resulting weight matrix.

Given the 2-dimensional form of the weight tensor, we can investigate the low-rank structure in the convolutional layers as for the dense layers. As noted already in Figure 4.5, we observe the same strong low-rank structure behaviour in the flattened convolutional layers as in the dense layers. Interestingly, the low-rank structure is the most visible in the final convolutional layers, which also contain the highest number of parameters, see Figure A.5.

Importantly, note that after performing the low-rank approximation in this 2-dimensional space, we can reshape the resulting 2-dimensional low-rank matrices back into the 4-dimensional form of a convolutional layer. Table A.1 shows that such a low-rank approximation of the convolutional layers of the analyzed ResNet-18 model can be performed without a loss in the model's predictive performance, while significantly reducing the total number of model parameters.

Method	-ELBO ↓	NLL ↓	Accuracy ↑	#Params ↓	%Params ↓
Mean-field	$122.61_{\pm 0.012}$	$0.495_{\pm 0.0080}$	$83.5_{\pm 0.37}$	9,814,026	100.0
1-tied	$122.57_{\pm 0.012}$	$0.658_{\pm 0.0069}$	$81.7_{\pm 0.39}$	4,929,711	50.2
2-tied	$122.77_{\pm 0.012}$	$0.503_{\pm 0.0080}$	$83.2_{\pm 0.37}$	4,946,964	50.4
3-tied	$122.67_{\pm 0.012}$	$0.501_{\pm 0.0079}$	$83.2_{\pm 0.37}$	4,964,217	50.6

Table A.1: Impact of the low-rank approximation of the GMFVI-trained posterior standard deviations of a ResNet-18 model on the model's predictive performance. We report mean and SEM of each metric across 100 weights samples. The low-rank approximations with ranks higher than one achieve predictive performance close to that when not using any approximations, while significantly reducing the number of model parameters.

A.4.4 k-tied Normal posterior training

To exploit the low-rank structure observation, we propose the k-tied Normal posterior, as discussed in Section 4.4. We study the properties of the k-tied Normal posterior applied to the MLP, CNN and LSTM models. We use the k-tied Normal variational posterior for all the dense layers of the analyzed models. Namely, we use the k-tied Normal variational posterior for all the three layers of the MLP model, for the two dense layers of the CNN model and for the LSTM cell's kernel and recurrent kernel.

We initialize the parameters u_{ik} and v_{jk} of the k-tied Normal distribution so that after the outer-product operation the respective standard deviations σ_{ij} have the same mean values as we obtain when using the standard GMFVI posterior parametrization. More precisely, we initialize the parameters u_{ik} and v_{jk} so that after the outer-product operation the respective σ_{ij} standard deviations have means at 0.01 before transforming to log-domain. This means that in the log domain the parameters u_{ik} and v_{jk} are initialized as $0.5(\log(0.01) - \log(k))$. We also add white noise $\mathcal{N}(0, 0.1)$ to the values of u_{ik} and v_{jk} in the log domain to break symmetry.

During training of the models with the k-tied Normal posterior, we linearly anneal the contribution of the D_{KL} term of the ELBO loss. We select the best linear coefficient for the annealing from $\{5 \times 10^{-5}, 5 \times 10^{-6}\}$ (per batch) and increase the effective contribution every

100 batches in a step-wise manner. In particular, we anneal the D_{KL} term to obtain the predictive performance results for all the models in Table 4.4. However, we do not perform the annealing in the Signal-to-Noise ratio (SNR) and negative ELBO convergence speed experiments in tables 4.5 and 4.6 respectively. In these two cases, KL annealing would occlude the values of interest, which show the clear impact of the k-tied Normal posterior.

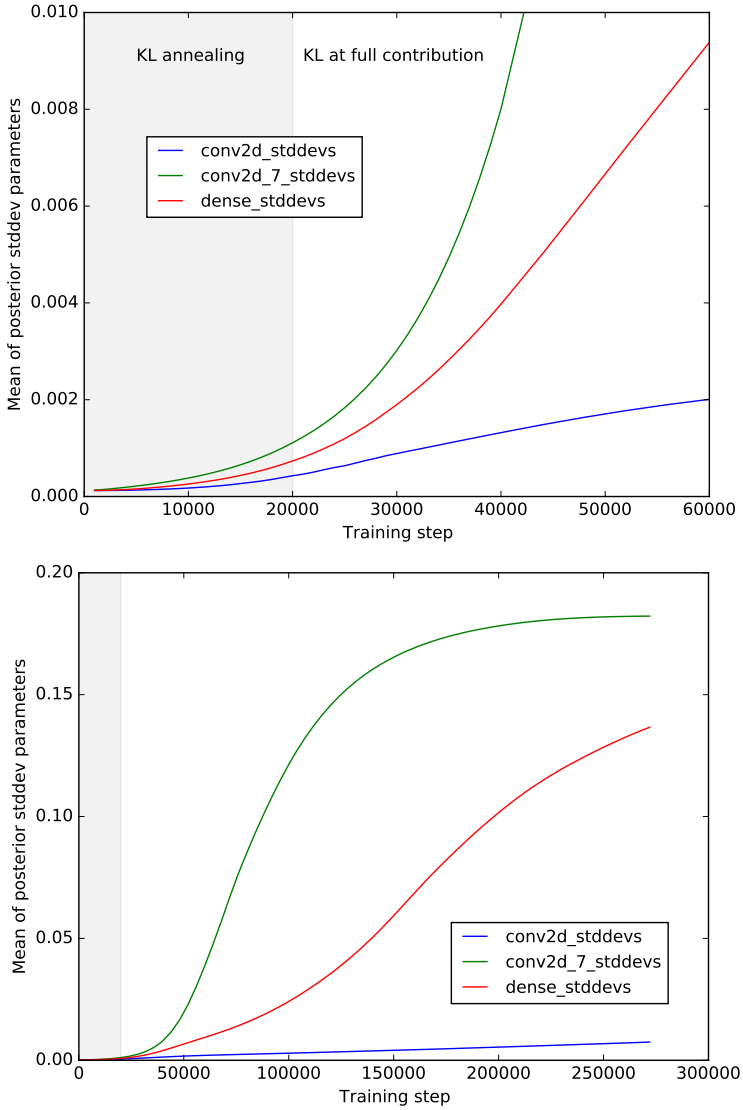


Figure A.3: Change in the mean of posterior standard deviation parameters for selected layers of the standard GMFVI ResNet-18 CIFAR-10 model over the course of training. KL is annealed over the first 50 epochs linearly from 0 to 1 (gray area). Top: posterior standard deviation parameters continue being optimized when the KL is at its full contribution. Bottom: the posterior standard deviations reach large values after 700 epochs showing that we are modeling substantial uncertainty.

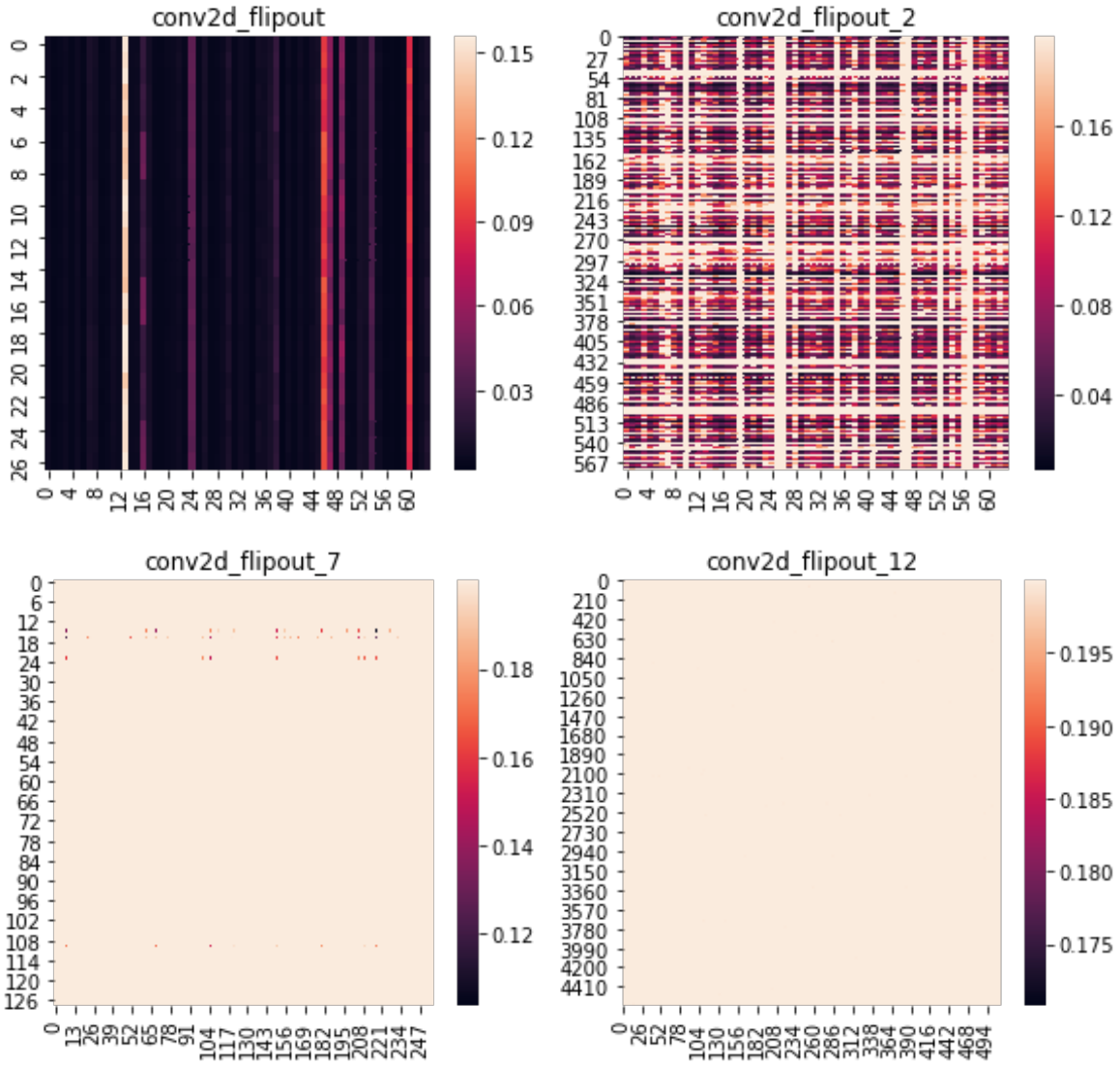


Figure A.4: Heat maps of the partially flattened posterior standard deviation tensors for the selected convolutional layers of the ResNet-18 GMFVI BNN trained on CIFAR-10. The partially flattened posterior standard deviation tensors of the convolutional layers display similar low-rank patterns that we observe for the dense layers.

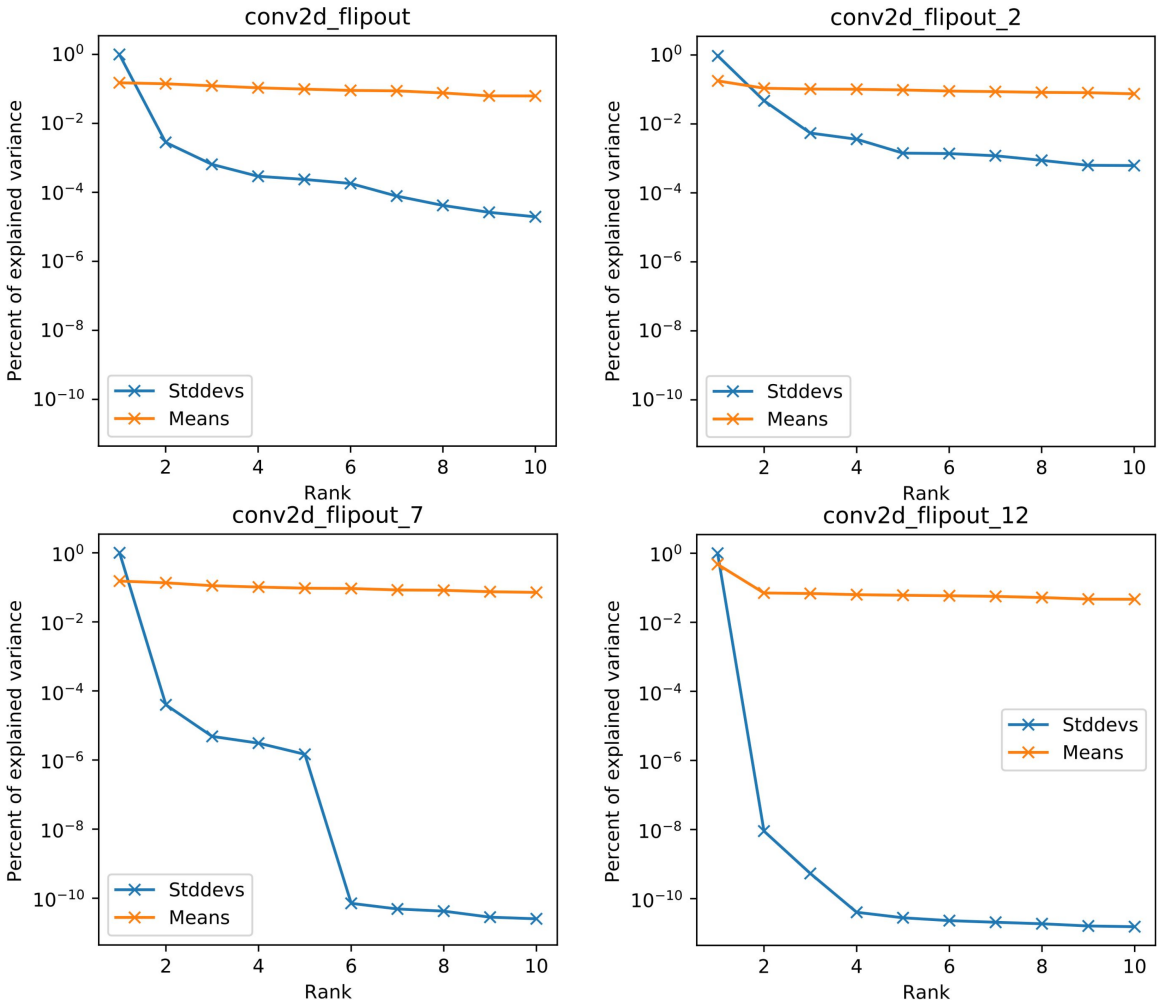


Figure A.5: Fraction of variance explained per each singular value from SVD of partially flattened tensors of posterior means and posterior standard deviations for different convolutional layers of the ResNet-18 GMFVI BNN trained on CIFAR-10. Posterior standard deviations clearly display strong low-rank structure, with most of the variance contained in the top few singular values, while this is not the case for posterior means. Interestingly, the low-rank structure is the most visible for the final convolutional layers, which also contain the highest number of parameters.

Appendix B

Appendix for Chapter 5

B.1 Model Details

We now give details regarding the models we use in all our experiments. We use Tensorflow version 2.1 and carry out all experiments on Nvidia P100 accelerators.

B.1.1 ResNet-20 CIFAR-10 Model

We use the CIFAR-10 dataset from (Krizhevsky et al., 2009a), in "version 3.0.0" provided in Tensorflow Datasets.[1] We use the Tensorflow Datasets training/testing split of 50,000 and 10,000 images, respectively.

We use the ResNet-20 model from `https://keras.io/examples/cifar10_resnet/` as a starting point. For our SGD baseline we use the exact same setup as in the Keras example (200 epochs, learning rate schedule, SGD with Nesterov acceleration). Notably the Keras example uses bias terms in all convolution layers, whereas some other implementations do not.

The Keras example page reports a reference test accuracy of 92.16 percent for the CIFAR-10 model, compared to our 92.22 percent accuracy. This is consistent with the larger literature, collected for example at `https://github.com/google/edward2/tree/master/baselines/cifar10`, with even higher accuracy achieved for variations of the ResNet model such as using wide layers, removing bias terms in the convolution layers, or additional regularization.

In Chapter 5, we study the phenomenon of poor $T = 1$ posteriors obtained by SG-MCMC and therefore use an accurate simulation and sampling setup at the cost of runtime. In order to obtain accurate simulations we use the following settings for SG-MCMC in every experiment, except where noted otherwise:

- Number of epochs: 1500
- Initial learning rate: $\ell = 0.1$
- Momentum decay: $\beta = 0.98$

[1] See `https://www.tensorflow.org/datasets/catalog/cifar10`

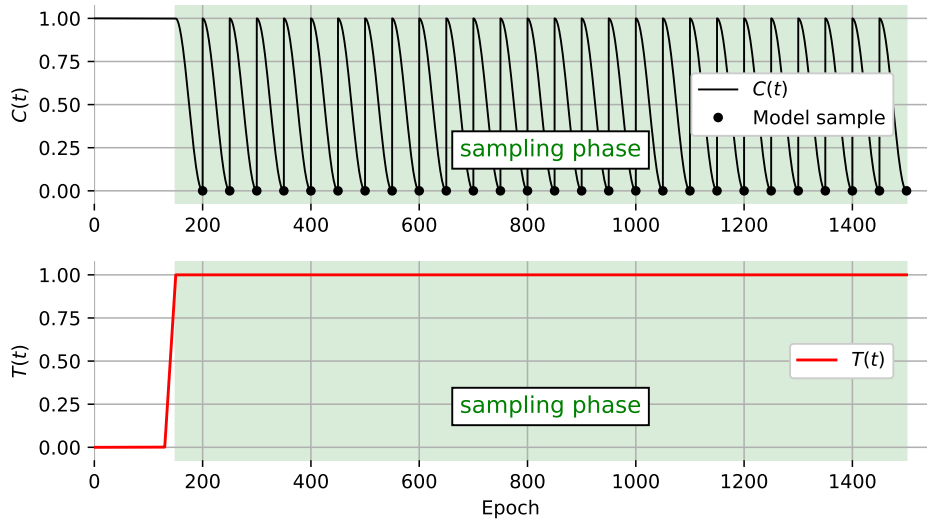


Figure B.1: Cyclical time stepping $C(t)$, and temperature ramp-up $T(t)$, as proposed by Zhang et al. (2019a) and used in Algorithm 1, for our ResNet-20 CIFAR-10 model (Section B.1.1). We sample one model at the end of each cycle when the inference accuracy is best, obtaining an ensemble of 27 models.

- Batch size: $|B| = 128$
- Sampling start: begin at epoch 150
- Cycle length: 50
- Cycle schedule: cosine
- Prior: $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$

For experiments on CIFAR-10 we use data augmentation as follows:

- random left/right flipping of the input image;
- border-padding by zero values, four pixels in horizontal and vertical direction, followed by a random cropping of the image to its original size.

We visualize the cyclic schedule used in our ResNet-20 CIFAR-10 experiments in Figure B.1.

B.1.2 ResNet-20 CIFAR-10 SGD Baseline

For the SGD baseline we follow the best practice from the existing Keras example which was tuned for generalization performance. In particular we use:

- Number of epochs: 200
- Initial learning rate: $\ell = 0.1$
- Momentum term: 0.9
- L2 regularization coefficient: 0.002
- Batch size: 128
- Optimizer: SGD with Nesterov momentum

- Learning rate schedule (epoch, ℓ-multiplier): $(80, 0.1)$, $(120, 0.01)$, $(160, 0.001)$, $(180, 0.0005)$.

Data augmentation is the same as described in Section B.1.1. We report the final validation performance and over the 200 epochs do not observe any overfitting.

B.1.3 CNN-LSTM IMDB Model

We use the IMDB sentiment classification text dataset provided by the `tensorflow.keras.datasets` API in Tensorflow version 2.1. We use 20,000 words and a maximum sequence length of 100 tokens. We use 20,000 training sequences and 25,000 testing sequences.

We use the CNN-LSTM example[2] as a starting point. For our SGD baseline we use the Keras model but add a prior $p(\theta) = \mathcal{N}(0, I)$ as used for the Bayesian posterior. We then use the Tensorflow SGD implementation to optimize the resulting $U(\theta)$ function. For SGD the model overfits and we therefore report the best end-of-epoch test accuracy and test cross-entropy achieved.

For all experiments, except where explicitly noted otherwise, we use the following parameters:

- Number of epochs: 500
- Initial learning rate: $\ell = 0.1$
- Momentum decay: $\beta = 0.98$
- Batch size: $|B| = 32$
- Sampling start: begin at epoch 50
- Cycle length: 25
- Cycle schedule: cosine
- Prior: $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$

We visualize the cyclic schedule used in our CNN-LSTM IMDB experiments in Figure B.2.

B.1.4 CNN-LSTM IMDB SGD Baseline

The SGD baseline follows the Keras example settings:

- Number of epochs: 50
- Initial learning rate: $\ell = 0.1$
- Momentum term: 0.98
- Regularization: MAP with $\mathcal{N}(0, I)$ prior
- Batch size: 32
- Optimizer: SGD with Nesterov momentum
- Learning rate schedule: None

We report the optimal test set performance from all end-of-epoch test evaluations. This is necessary because there is significant overfitting after the first ten epochs.

[2] Available at `https://github.com/keras-team/keras/blob/master/examples/imdb_cnn_lstm.py`

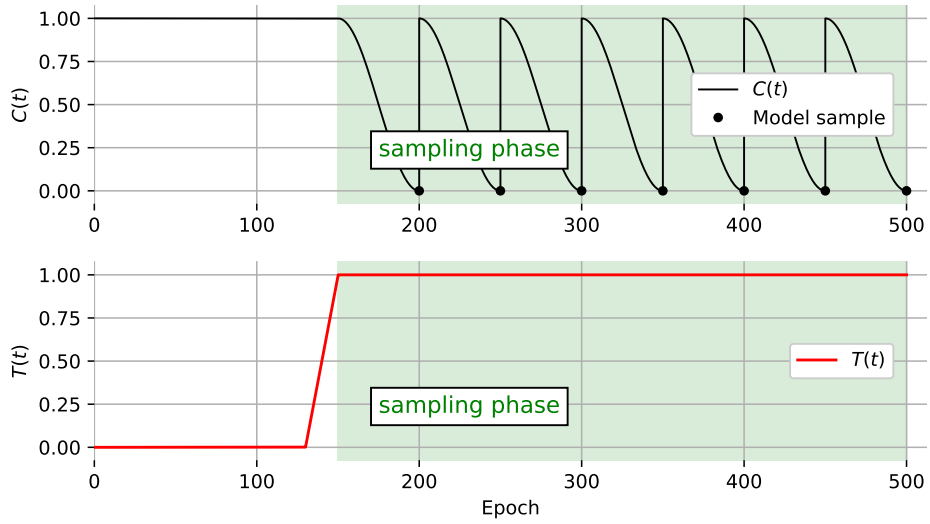


Figure B.2: Cyclical time stepping $C(t)$, and temperature ramp-up $T(t)$ for our CNN-LSTM IMDB model (Section B.1.3). We sample one model at the end of each cycle when the inference accuracy is best, obtaining an ensemble of 7 models.

B.2 Deep Learning Parameterization of SG-MCMC Methods

We derive the bijection between (learning rate ℓ, momentum decay β) and (timestep h, friction γ) by considering the *instantaneous gradient effect* α on the parameter, i.e. the amount by which the current gradient at time t affects the current gradient update update at time t. We set $\alpha = \ell/n$, where ℓ is the familiar learning rate parameter used in SGD and the factor $1/n$ is to convert $\nabla_{\boldsymbol{\theta}} U$ to $\nabla_{\boldsymbol{\theta}} G$, as $\nabla_{\boldsymbol{\theta}} G = \nabla_{\boldsymbol{\theta}} U/n$ is the familiar minibatch mean gradient. Likewise, the *momentum decay* is the factor $\beta < 1$ by which the momentum vector $\mathbf{m}^{(t)}$ is shrunk in each discretized time step. Having determined α and β we can derive two non-linear equations that depend on the particular time discretization used; for the symplectic Euler Langevin scheme these are

$$h^2 = \alpha \quad \left(= \frac{\ell}{n}\right), \qquad \text{and} \qquad 1 - h\gamma = \beta. \tag{B.1}$$

Solving these equations for h and γ simultaneously, given ℓ, n, and β yields the bijection

$$h = \sqrt{\ell/n}, \tag{B.2}$$
$$\gamma = (1-\beta)\sqrt{n/\ell}. \tag{B.3}$$

B.3 Connection to Stochastic Gradient Descent (SGD)

We now give a precise connection between stochastic gradient descent (SGD) and the symplectic Euler SG-MCMC method, Algorithm 1.

Algorithm 2: Stochastic Gradient Descent with Momentum (SGD) in Tensorflow.

1 **Function** `SGD`$(\tilde{G},\, \boldsymbol{\theta}^{(0)},\, \ell,\, \beta)$
  Input: $\tilde{G} : \Theta \to \mathbb{R}$ average batch loss function, cf equation (2.16); $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$ initial parameter; $\ell > 0$ learning rate parameter; $\beta \in [0, 1)$ momentum decay parameter.
  Output: Parameter sequence $\boldsymbol{\theta}^{(t)}$, at step $t = 1, 2, \ldots$
2 $\mathbf{m}^{(0)} \leftarrow \mathbf{0}$ `// Initialize momentum`
3 **for** $t = 1, 2, \ldots$ **do**
4 $\mathbf{m}^{(t)} \leftarrow \beta\, \mathbf{m}^{(t-1)} - \ell\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)})$ `// Update momentum`
5 $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \mathbf{m}^{(t)}$ `// Update parameters`
6 **yield** $\boldsymbol{\theta}^{(t)}$ `// Parameter at step` t

Algorithm 2 gives the stochastic gradient descent (SGD) with momentum algorithm as implemented in *Tensorflow*'s version 2.1 optimization methods, `tensorflow.keras.optimizers.SGD` and `tensorflow.train.MomentumOptimizer`, (Abadi et al., 2016).

Starting with Algorithm 2 we first perform an equivalent substitution of the moments,

$$\tilde{\mathbf{m}}^{(t)} := \sqrt{\frac{n}{\ell}}\, \mathbf{m}^{(t)}, \qquad \text{respectively,} \tag{B.4}$$

$$\mathbf{m}^{(t)} := \sqrt{\frac{\ell}{n}}\, \tilde{\mathbf{m}}^{(t)}, \tag{B.5}$$

we obtain the update from line 4 in Algorithm 2,

$$\sqrt{\frac{\ell}{n}}\, \tilde{\mathbf{m}}^{(t)} \leftarrow \beta \sqrt{\frac{\ell}{n}}\, \tilde{\mathbf{m}}^{(t-1)} - \ell\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)}. \tag{B.6}$$

Multiplying both sides of (B.6) by $\sqrt{n}/\sqrt{\ell}$ we obtain an equivalent form of Algorithm 2 with lines 4 and 5 replaced by

$$\tilde{\mathbf{m}}^{(t)} \leftarrow \beta\, \tilde{\mathbf{m}}^{(t-1)} - \sqrt{\ell n}\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)}), \tag{B.7}$$

$$\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \sqrt{\frac{\ell}{n}}\, \tilde{\mathbf{m}}^{(t)}. \tag{B.8}$$

From the bijection (B.2–B.3) we have $h = \sqrt{\ell/n}$ and $\gamma = (1 - \beta)\sqrt{n/\ell}$. Solving for β gives

$$\beta = 1 - \gamma \sqrt{\frac{\ell}{n}} = 1 - \gamma h. \tag{B.9}$$

We also have

$$\sqrt{\ell n} = \sqrt{\frac{\ell}{n} n^2} = n \sqrt{\frac{\ell}{n}} = h\, n. \tag{B.10}$$

Algorithm 3: Stochastic Gradient Descent with Momentum (SGD), reparameterized.

1 **Function** `SGDEquivalent`($\tilde{G}$, $\boldsymbol{\theta}^{(0)}$, ℓ, β)
  Input: $\tilde{G} : \Theta \to \mathbb{R}$ average batch loss function, cf equation (2.16); $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$ initial parameter; $h > 0$ discretization step size parameter; $\gamma > 0$ friction parameter.
  Output: Parameter sequence $\boldsymbol{\theta}^{(t)}$, $t = 1, 2, \ldots$, at step t
2 $\tilde{\mathbf{m}}^{(0)} \leftarrow \mathbf{0}$ `// Initialize momentum`
3 **for** $t = 1, 2, \ldots$ **do**
4 $\tilde{\mathbf{m}}^{(t)} \leftarrow (1 - \gamma h)\, \tilde{\mathbf{m}}^{(t-1)} - h\, n\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)})$ `// Update momentum`
5 $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + h\, \tilde{\mathbf{m}}^{(t)}$ `// Update parameters`
6 **yield** $\boldsymbol{\theta}^{(t)}$ `// Parameter at step` t

Substituting (B.9) and (B.10) into (B.7) and (B.8) gives the equivalent updates

$$\tilde{\mathbf{m}}^{(t)} \leftarrow (1 - \gamma h)\, \tilde{\mathbf{m}}^{(t-1)} - h\, n\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)}), \tag{B.11}$$

$$\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + h\, \tilde{\mathbf{m}}^{(t)}. \tag{B.12}$$

These equivalent changes produce Algorithm 3. Algorithm 2 and Algorithm 3 generate equivalent trajectories $\boldsymbol{\theta}^{(t)}$, $t = 1, 2, \ldots$, but differ in the scaling of their momenta, $\mathbf{m}^{(t)}$ and $\tilde{\mathbf{m}}^{(t)}$.

Comparing lines 4–5 in Algorithm 3 with lines 13–14 in Algorithm 1 we see that when $\mathbf{M} = I$ and $C(t) = 1$ the only remaining difference between the updates is the additional noise $\sqrt{2\gamma h T}\, \mathbf{M}^{1/2} \mathbf{R}^{(t)}$ in the SG-MCMC method. In this *precise* sense the SG-MCMC Algorithm 1 is just "SGD with noise".

B.4 Semi-Adaptive Estimation of Layerwise Preconditioner M

During our experiments with deep learning models we noticed that both minibatch noise as well as gradient magnitudes tend to behave similar within a set of related parameters. For example, for a given learning iteration, all gradients related to convolution kernel weights of the same convolution layer of a network tend to have similar magnitudes and minibatch noise variance. At the same iteration they may be different from the magnitudes and minibatch noise variance of gradients of the parameters of another layer in the same network.

Therefore, we estimate a simple diagonal preconditioner that ties together the scale of all parameter elements that belong to the same model variable. Moreover, we normalize the preconditioner so that the least sensitive variable always has scale one. With such normalization, if all variables would be equally sensitive the preconditioner becomes $\mathbf{M} = I$, the identity preconditioner.

We estimate the layerwise preconditioner using Algorithm 4.

Updating the preconditioner. In Langevin schemes the preconditioner couples the moment space to the parameter space. If we use a new estimate $\mathbf{M}'$ to replace the old

Algorithm 4: Estimate Layerwise Preconditioner.

Function `EstimateM`($\tilde{G}$, $\boldsymbol{\theta}$, K, ϵ)

Input: $\tilde{G} : \Theta \to \mathbb{R}$ mean energy function estimate; $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_S) \in \mathbb{R}^{d_1 \times \cdots \times d_S}$ current model parameter variables; K number of minibatches (default $K = 32$); ϵ regularization value (default $\epsilon = 10^{-7}$)

Output: Preconditioning matrix $\mathbf{M}$

for $s = 1, 2, \ldots, S$ **do**
  $\mathbf{v}_s \leftarrow \mathbf{0}$
for $k = 1, 2, \ldots, K$ **do**
  $\mathbf{g}^{(k)} \leftarrow \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta})$ `// Noisy gradient`
  for $s = 1, 2, \ldots, S$ **do**
    $\mathbf{v}_s \leftarrow \mathbf{v}_s + \mathbf{g}_s^{(k)} \cdot \mathbf{g}_s^{(k)}$
for $s = 1, 2, \ldots, S$ **do**
  $\sigma_s \leftarrow \sqrt{\epsilon + \frac{1}{d_s K} \sum_i \mathbf{v}_{s,i}}$ `// RMSprop`
$\sigma_{\min} \leftarrow \min_s \sigma_s$ `// Least sensitive`
for $s = 1, 2, \ldots, S$ **do**
  $\mathbf{M}_s \leftarrow \frac{\sigma_s}{\sigma_{\min}} I$
$\mathbf{M} \leftarrow \begin{bmatrix} \mathbf{M}_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \mathbf{M}_S \end{bmatrix}$
return $\mathbf{M}$

preconditioner $\mathbf{M}$ then we change this coupling and if left unchanged then the old moments $\mathbf{m}$ would no longer have the correct distribution.[3] We therefore posit that upon changing the preconditioner the effect of the moments should remain the same. To retain the full information in the current moments we set $\mathbf{m}' = \mathbf{M}'^{1/2}\mathbf{M}^{-1/2}\mathbf{m}$ which we can understand as $\mathbf{M}'^{1/2}(\mathbf{M}^{-1/2}\mathbf{m})$, where the bracketed part canonicalizes the moments $\mathbf{m}$ to the identity preconditioner, and $\mathbf{M}'^{1/2}$ transfers the canonical moments to the new preconditioner.

B.5 Kullback-Leibler Scaling in Variational Bayesian Neural Networks

With the posterior energy $U(\boldsymbol{\theta})$ defined in Chapter 5, we define two variants of tempered posterior energies:

- Fully tempered energy: $U_F(\boldsymbol{\theta}) = U(\boldsymbol{\theta})/T$, and
- Partially tempered energy: $U_P(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}) - \frac{1}{T}\sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta})$.

[3] More precisely, $\mathbf{M}^{-1/2}\mathbf{m}$ should always be distributed according to $\mathcal{N}(0, I)$.

Note that $U_F(\boldsymbol{\theta})$ is used for all experiments in Chapter 5 and temper both the log-likelihood as well as the log-prior terms, whereas $U_P(\boldsymbol{\theta})$ only scales the log-likelihood terms while leaving the log-prior untouched.

We now show that Kullback-Leibler scaling as commonly done in variational Bayesian neural networks corresponds to approximating the partially tempered posterior,

$$p_P(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-U_P(\boldsymbol{\theta})). \tag{B.13}$$

For any distribution $q(\boldsymbol{\theta})$ we consider the Kullback-Leibler divergence,

$$\begin{aligned} &D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p_P(\boldsymbol{\theta}|\mathcal{D})) && \text{(B.14)}\\ &= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log p_P(\boldsymbol{\theta}|\mathcal{D})\right] && \text{(B.15)}\\ &= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log \frac{\exp(-U_P(\boldsymbol{\theta}))}{\int \exp(-U_P(\boldsymbol{\theta}'))\,\mathrm{d}\boldsymbol{\theta}'}\right]. && \text{(B.16)} \end{aligned}$$

The normalizing integral in (B.16) is not a function of $\boldsymbol{\theta}$ and thus does not depend on $q(\boldsymbol{\theta})$, allowing us to simplify the equation further:

$$\begin{aligned} &= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) - \frac{1}{T}\sum_{i=1}^{n} \log p(y_i|x_i,\boldsymbol{\theta})\right] && \text{(B.17)}\\ &\qquad + \underbrace{\log \int \exp(-U_P(\boldsymbol{\theta}))\,\mathrm{d}\boldsymbol{\theta}}_{\text{constant, } =: \log E_P} && \text{(B.18)}\\ &= D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \frac{1}{T}\sum_{i=1}^{n} \log p(y_i|x_i,\boldsymbol{\theta}) + \log E_P. && \text{(B.19)} \end{aligned}$$

Here we defined E_P as the *partial temperized evidence* which does not depend on $\boldsymbol{\theta}$ and therefore becomes a constant. The global minimizer of (B.19) over all distributions $q \in \mathcal{Q}$ is the unique distribution $p_P(\boldsymbol{\theta}|\mathcal{D})$, (MacKay et al., 1995).

We now consider this minimizer, substituting $\lambda := T$,

$$\begin{aligned} &\operatorname*{argmin}_{q\in\mathcal{Q}} D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p_P(\boldsymbol{\theta}|\mathcal{D})) && \text{(B.20)}\\ &= \operatorname*{argmin}_{q\in\mathcal{Q}} D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \frac{1}{T}\sum_{i=1}^{n} \log p(y_i|x_i,\boldsymbol{\theta}) && \text{(B.21)} \end{aligned}$$

The minimizing $q \in \mathcal{Q}$ does not depend on the overall scaling of the optimizing function. We can therefore scale the function by a factor of T,

$$= \operatorname*{argmin}_{q\in\mathcal{Q}} T D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \sum_{i=1}^{n} \log p(y_i|x_i,\boldsymbol{\theta}) \tag{B.22}$$

Substituting $\lambda := T$ yields

$$= \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \; \lambda D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \sum_{i=1}^{n} \log p(y_i | x_i, \boldsymbol{\theta}). \tag{B.23}$$

The last equation, (B.23) is the KL-weighted negative evidence lower bound (ELBO) objective commonly used in variational Bayes for Bayesian neural networks, confer the ELBO equation (5.1).

B.6 Inference Bias-Variance Trade-off Hypothesis

Bias-variance Tradeoff Hypothesis: For $T = 1$ the posterior is diverse and there is high variance between model predictions. For $T \ll 1$ we sample nearby modes and reduce prediction variance but increase bias; the variance dominates the error and reducing variance ($T \ll 1$) improves predictive performance.

We approach the hypothesis using a simple asymptotic argument. We consider the SG-MCMC method we use, including preconditioning and cyclical time stepping. Whereas within a cycle the Markov chain is non-homogeneous, if we consider only the end-of-cycle iterates that emit a parameter $\boldsymbol{\theta}^{(t)}$, then this coarse-grained process is a homogeneous Markov chain. For such Markov chains we can leverage generalized central limit theorems for functions of $\boldsymbol{\theta}$, see e.g. (Jones et al., 2004; Häggström & Rosenthal, 2007), and because of existence of limits we can consider the asymptotic behavior of the test cross-entropy performance measure $C(S)$ as we increase the ensemble size $S \to \infty$.

In particular, expectations of smooth functions of empirical means of S samples have an expansion of the form, (Nowozin, 2018; Schucany et al., 1971),

$$\mathbb{E}[C(S)] = C(\infty) + a_1 \frac{1}{S} + a_2 \frac{1}{S^2} + \dots. \tag{B.24}$$

Risk Asymptotics Experiment: if we can estimate $C(\infty)$ we know what performance we could achieve if we were to keep sampling. To this end we apply a simple linear regression estimate, (Schucany et al., 1971), to the empirically observed performance estimates $\hat{C}(S)$ for different ensemble sizes S. By truncation at second order, we obtain estimates for $C(\infty)$, a_1, and a_2.

In Figure B.3 we show the regressed test cross-entropy metric obtained by fitting (B.24) to second order to all samples for $S \geq 20$ close to the asymptotic regime, and visualize the estimate $\hat{C}(\infty)$. In Figure B.4 we visualize our estimated $\hat{C}(\infty)$ as a function of the temperature T. The results indicate two things: *first*, we could gain better predictive performance from running our SG-MCMC method for longer (Figure B.3); but *second*, the additional gain that could be obtained from longer sampling is too small to make $T = 1$ superior to $T < 1$ (Figure B.4).

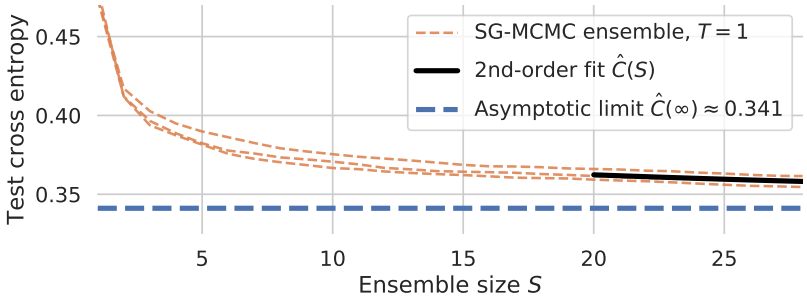


Figure B.3: Regressing the limiting ResNet-20/CIFAR-10 ensemble performance: at temperature $T = 1$ an ensemble of size $S = \infty$ would achieve 0.341 test cross-entropy. For SG-MCMC we show three different runs with varying seeds.

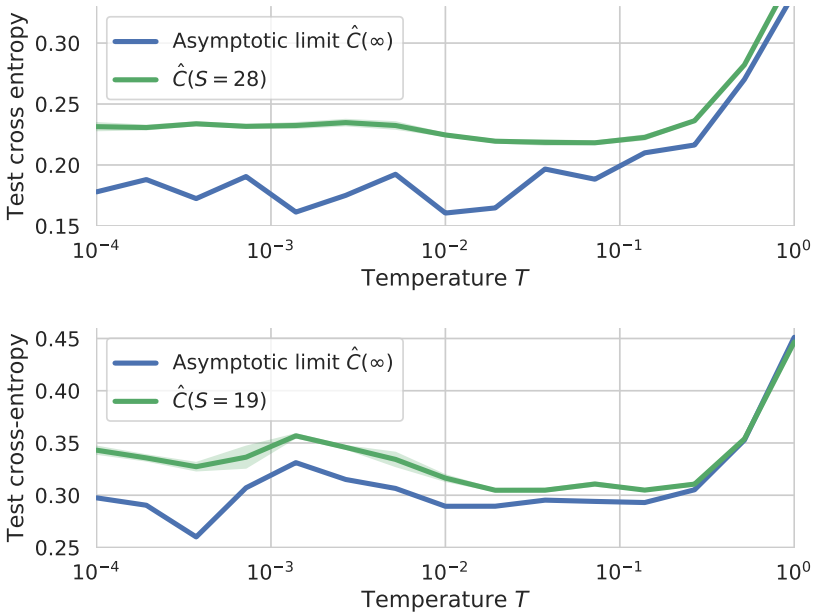


Figure B.4: Ensemble variance for ResNet-20/CIFAR-10 (**top**) and CNN-LSTM/IMDB (**bottom**) does not explain poor performance at $T = 1$: even in the infinite limit the performance $C(\infty)$ remains poor compared to $T < 1$.

B.7 Cold posteriors improve uncertainty metrics.

In Chapter 5, we show that cold posteriors improve prediction performance in terms of accuracy and cross entropy. Figure B.5 and Figure B.6 show that for both the ResNet-20 and the CNN-LSTM model, cold posteriors also improve the uncertainty metrics Brier score Brier (1950) and expected calibration error (ECE) Naeini et al. (2015).

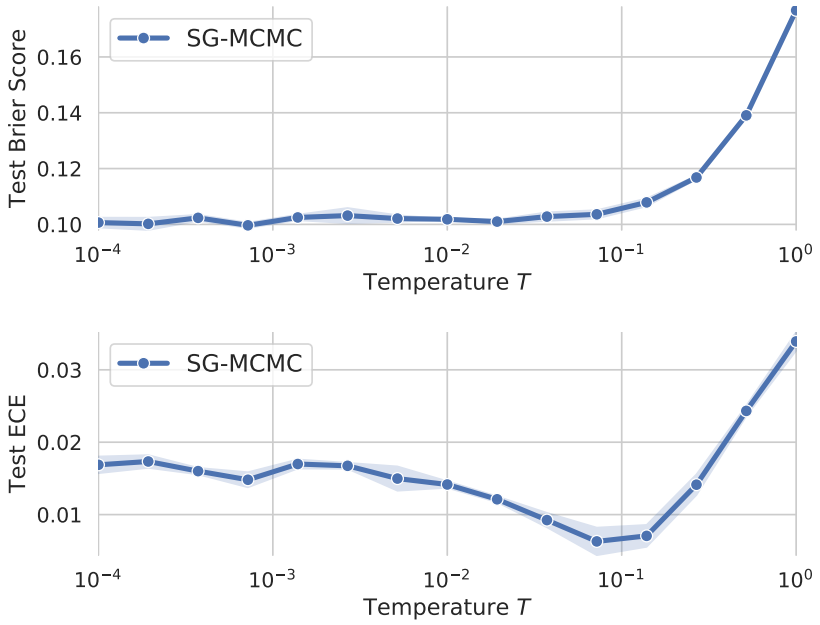


Figure B.5: ResNet-20/CIFAR-10: Chapter 5, we show that cold posteriors improve prediction performance in terms of accuracy and cross entropy (Figure 5.1 and Figure 5.2). This plot shows that cold posteriors also improve the uncertainty metrics Brier score and expected calibration error (ECE) (lower is better).

B.8 Details on the Experiment for the Implicit Initialization Prior in SGD Hypothesis

SGD and SG-MCMC are set up as described in Appendix B.1.1. In Chapter 5, the test accuracy as a function of epochs is shown in Figure 5.13. In Figure B.7 we additionally report the test cross entropy for the same experiment. SGD initialized by the last model of the SG-MCMC sampling dynamics also recovers the same performance in terms of cross entropy as vanilla SGD.

B.9 Diagnostics: Temperatures

The following proposition adapted from (Leimkuhler & Matthews, 2016, Section 6.1.5) provides a general way to construct temperature observables.

Proposition 1 (Constructing Temperature Observables)**.** *Given a Hamiltonian* $H(\boldsymbol{\theta}, \mathbf{m})$ *corresponding to Langevin dynamics,*

$$H(\boldsymbol{\theta}, \mathbf{m}) = \frac{1}{T} U(\boldsymbol{\theta}) + \frac{1}{2} \mathbf{m}^T \mathbf{M}^{-1} \mathbf{m}, \tag{B.25}$$

and an arbitrary smooth vector field $B : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$ *satisfying*

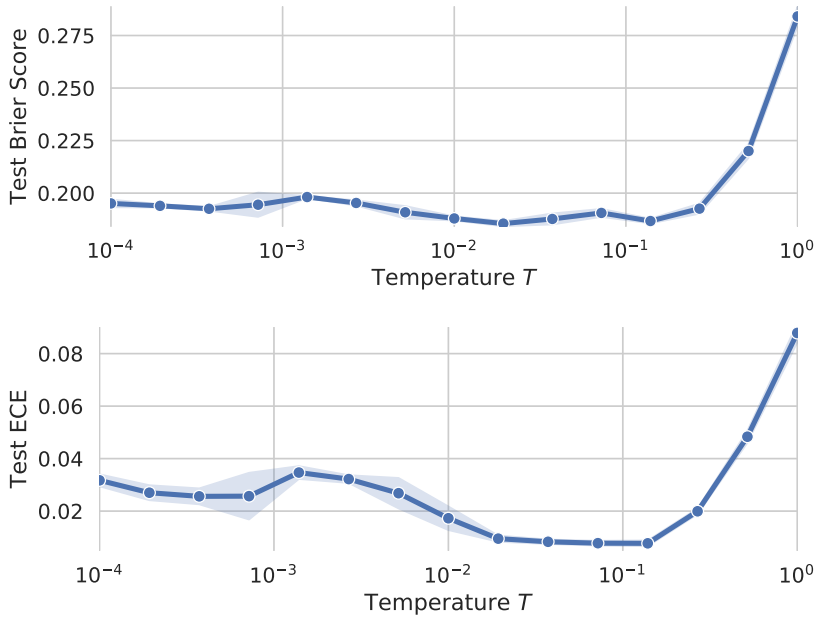


Figure B.6: CNN-LSTM/IMDB: Cold posteriors also improve the uncertainty metrics Brier score and expected calibration error (ECE) (lower is better). The plots for accuracy and cross entropy are shown in Figure 5.3.

- $0 < \mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle B(\boldsymbol{\theta},\mathbf{m}), \nabla H(\boldsymbol{\theta},\mathbf{m})\rangle] < \infty$,
- $0 < \mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle \mathbf{1}_{2d}, \nabla B(\boldsymbol{\theta},\mathbf{m})\rangle] < \infty$, *and*
- $\|B(\boldsymbol{\theta},\mathbf{m})\ \exp(-H(\boldsymbol{\theta},\mathbf{m}))\| < \infty$ *for all* $(\boldsymbol{\theta},\mathbf{m}) \in \mathbb{R}^d \times \mathbb{R}^d$,

then

$$T = \frac{\mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle B(\boldsymbol{\theta},\mathbf{m}), \nabla H(\boldsymbol{\theta},\mathbf{m})\rangle]}{\mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle \mathbf{1}_{2d}, \nabla B(\boldsymbol{\theta},\mathbf{m})\rangle]}. \tag{B.26}$$

Note that for the Hamiltonian (B.25) we have, assuming a symmetric preconditioner, $(\mathbf{M}^{-1})^T = \mathbf{M}^{-1}$,

$$\nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta},\mathbf{m}) = \frac{1}{T}\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}), \tag{B.27}$$

$$\nabla_{\mathbf{m}} H(\boldsymbol{\theta},\mathbf{m}) = \mathbf{M}^{-1}\,\mathbf{m}. \tag{B.28}$$

B.9.1 Kinetic Temperature Estimation

Simulating the Langevin dynamics, equations (2.14–2.15), produces moments $\mathbf{m}$ which are jointly distributed according to a multivariate Normal distribution, (Leimkuhler & Matthews, 2016),

$$\mathbf{m} \sim \mathcal{N}(0, \mathbf{M}). \tag{B.29}$$

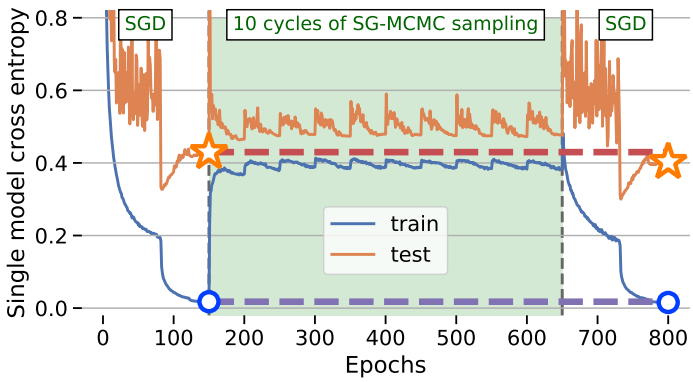


Figure B.7: Do the SG-MCMC dynamics harm a beneficial initialization bias used by SGD? We first train a ResNet-20 on CIFAR-10 via SGD, then switch over to SG-MCMC sampling and finally switch back to SGD optimization. We report the single-model test cross entropy of SGD and the SG-MCMC chain as function of epochs. SGD recovers from being initialized by the SG-MCMC state.

The *kinetic temperature* $\hat{T}_K(\mathbf{m})$ is derived from the moments as

$$\hat{T}_K(\mathbf{m}) := \frac{\mathbf{m}^T \mathbf{M}^{-1} \mathbf{m}}{d}, \tag{B.30}$$

and we have that for a perfect simulation of the dynamics we achieve $\mathbb{E}[\hat{T}_K(\mathbf{m})] = T$, where T is the target temperature of the system, (Leimkuhler & Matthews, 2016). This can be seen by instantiating Proposition 1 for the Langevin Hamiltonian and $B_K(\boldsymbol{\theta}, \mathbf{m}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{m} \end{bmatrix}$.

In general we only approximately solve the SDE and errors in the solution arise due to discretization, minibatch noise, or lack of full equilibration to the stationary distribution. Therefore, we can use $\hat{T}_K(\mathbf{m})$ as a diagnostic to measure the temperature of the current system state, and a deviation from the target temperature could diagnose poor solution accuracy. To this end, we know that if $\mathbf{m} \sim \mathcal{N}(0, \mathbf{M})$ then $(\mathbf{M}^{-1/2}\,\mathbf{m}) \sim \mathcal{N}(0, I_d)$ and thus the inner product $(\mathbf{M}^{-1/2}\,\mathbf{m})^T\,(\mathbf{M}^{-1/2}\,\mathbf{m}) = \mathbf{m}^T\,\mathbf{M}^{-1}\,\mathbf{m}$ is distributed according to a standard χ^2-distribution with d degrees of freedom,

$$(\mathbf{m}^T \mathbf{M}^{-1} \mathbf{m}) \sim \chi^2(d). \tag{B.31}$$

The $\chi^2(d)$ distribution has mean d and variance $2d$ and we can use the tail probabilities to test whether the observed temperature could arise from an accurate discretization of the SDE (2.14–2.15). For a given confidence level $c \in (0, 1)$, e.g. $c = 0.99$, we define the confidence interval

$$J_{T_K}(d, c) := \left(\frac{T}{d} F^{-1}_{\chi^2(d)} \left(\frac{1-c}{2} \right), \frac{T}{d} F^{-1}_{\chi^2(d)} \left(\frac{1+c}{2} \right) \right), \tag{B.32}$$

where $F^{-1}_{\chi^2(d)}$ is the inverse cumulative distribution function of the χ^2 distribution with d degrees of freedom. By construction if (B.31) holds, then $\hat{T}_K(\mathbf{m}) \in J_{T_K}(d, c)$ with probability c exactly.

Therefore, if c is close to one, say $c = 0.99$, and we find that $\hat{T}_K(\mathbf{m}) \notin J(d, c)$ this indicates issues of discretization error or convergence of the SDE (2.14–2.15).

Because (B.29) holds for any subvector of $\mathbf{m}$, we can create one kinetic temperature estimate for each model variable separately, such as one or two scalar temperature estimates for each layer (e.g. one for the weights and one for the bias of a `Dense` layer). We found per-layer temperature estimates helpful in diagnosing convergence issues and this directly led to the creation of our layerwise preconditioner.

B.9.2 Configurational Temperature Estimation

The so called *configurational temperature*[4] is defined as

$$\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})) = \frac{\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) \rangle}{d}. \tag{B.33}$$

For a perfect simulation of SDE (2.14–2.15) we have $\mathbb{E}[\hat{T}_C] = T$, where T is the target temperature of the system. This can be seen by instantiating Proposition 1 for the Langevin Hamiltonian and $B_C(\boldsymbol{\theta}, \mathbf{m}) = \begin{bmatrix} \boldsymbol{\theta} \\ \mathbf{0} \end{bmatrix}$.

As for the kinetic temperature diagnostic, we can instantiate Proposition 1 for arbitrary subsets of parameters by a suitable choice of $B_C(\boldsymbol{\theta}, \mathbf{m})$. However, whereas for the kinetic temperature the exact sampling distribution of the estimate is known in the form of a scaled χ^2 distribution, we are not aware of a characterization of the sampling distribution of configurational temperature estimates. It is likely this sampling distribution depends on $U(\boldsymbol{\theta})$ and thus does not have a simple form. Proposition 1 only asserts that under the true target distribution we have

$$\mathbb{E}_{\boldsymbol{\theta} \sim \exp(-U(\boldsymbol{\theta})/T)}[\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}))] = T. \tag{B.34}$$

Because (B.33) is the empirical average of per parameter random variables, if all these variables have finite variance the central limit theorem asserts that for large d we can expect

$$\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})) \sim \mathcal{N}(T, \sigma^2_{T_C}), \tag{B.35}$$

with unknown variance $\sigma^2_{T_C}$.

Recent work of Yaida (2018) provides a similar diagnostic, equation (FDR1') in their work, to the configurational temperature (B.33) for the SGD equilibrium distribution under finite time dynamics. However, our goal here is different: whereas Yaida (2018) is interested in diagnosing convergence to the SGD equilibrium distribution in order to adjust learning rates we instead want to diagnose discrepancy of our current dynamics against the true target distribution.

[4] Sometimes other quantities are also refered to as configurational temperature, see (Leimkuhler & Matthews, 2016, Section 6.1.5).

B.10 Simulation Accuracy Ablation Study

Equipped with the diagnostics of Section B.9 we can now study how accurate our algorithms simulate the Langevin dynamics. We will demonstrate that layerwise preconditioning and cyclical time stepping are individually effective at improving simulation accuracy, however, only by combining these two methods we can achieve high simulation accuracy on the CNN-LSTM model as measured by our diagnostics.

Setup. We perform the same ResNet-20 CIFAR-10 and CNN-LSTM IMDB experiments as in Chapter 5, but consider four variations of our algorithm: with and without preconditioning, and with and without cosine time stepping schedules. In case no preconditioner is used we simply set $\mathbf{M} = I$ for all iterations. In case no cosine time stepping is used we simply set $C(t) = 1$ for all iterations.

Independent of whether cosine time stepping is used we divide the iterations into cycles and for each method consider all models at the end of a cycle, where we hope simulation accuracy is the highest. We then evaluate the temperature diagnostics for all model variables. For the kinetic temperatures, if simulation is accurate then 99 percent of the variables should on average lie in the 99% high probability region under the sampling distribution. For the configurational temperature we can only report the average configurational temperature across all the end-of-cycle models.

Results. We report the results in Table B.1 and Table B.2 and visualize the kinetic temperatures in Figures B.8 to B.11 and Figures B.12a to B.12d.

The results indicate that both cosine time stepping and layerwise preconditioning have a beneficial effect on simulation accuracy. For ResNet-20 cyclical time stepping is sufficient for high simulation accuracy, but it is by itself not able to achieve high accuracy on the CNN-LSTM model. For both models the combination of cyclical time stepping and preconditioning (Figure B.8 and Figure B.12a) achieves a high simulation accuracy, that is, all kinetic temperatures match the sampling distribution of the Langevin dynamics, indicating—at least with respect to the power of our diagnostics—accurate simulation.

Another interesting observation can be seen in Table B.1: we can achieve a high accuracy of ≥ 88 percent even in cases where the simulation accuracy is poor. This indicates that optimization is different from accurate Langevin dynamics simulation.

B.11 Dirty Likelihood Functions

Dirty Likelihood Hypothesis: Deep learning practices that violate the likelihood principle (batch normalization, dropout, data augmentation) cause deviation from the Bayes posterior.

We now discuss how batch normalization, dropout, and data augmentation produce non-trivial modifications to the likelihood function. We call the resulting likelihood functions

Precond	Cyclic	$\hat{\mathbb{E}}[\hat{T}_K \in \mathcal{R}_{99}]$	$\hat{\mathbb{E}}[\hat{T}_C]$	Accuracy (%)	Cross-entropy
✓	✓	0.989±0.0014	0.94±0.011	88.2±0.11	0.358±0.0011
✗	✓	0.9772±0.00059	1.02±0.018	88.49±0.014	0.3500±0.00064
✓	✗	0.905±0.0019	1.23±0.046	88.0±0.10	0.3808±0.00064
✗	✗	0.676±0.0052	1.7±0.18	86.86±0.072	0.507±0.0080

Table B.1: ResNet-20 CIFAR-10 simulation accuracy ablation at $T = 1$: layerwise preconditioning and cyclical time stepping each have a beneficial effect on improving inference accuracy and the effect is complementary. $\hat{\mathbb{E}}[\hat{T}_K \in \mathcal{R}_{99}]$ is the empirically estimated probability that the kinetic temperature statistics are in the 99% confidence interval, the ideal value is 0.99. $\hat{\mathbb{E}}[\hat{T}_C]$ is the empirical average of the configurational temperature estimates, the ideal value is 1.0. For both quantities we take the value achieved at the end of each cycle, that is, whenever $C(t) = 0$ and average all the resulting values. The deviation is given in ±SEM where SEM is the standard error of the mean estimated from three independent experiment replicates. Both preconditioning and cyclical time stepping are effective at improving the simulation accuracy.

Precond	Cyclic	$\hat{\mathbb{E}}[\hat{T}_K \in \mathcal{R}_{99}]$	$\hat{\mathbb{E}}[\hat{T}_C]$	Accuracy (%)	Cross-entropy
✓	✓	0.954±0.0053	0.99122±0.000079	81.95±0.22	0.425±0.0032
✗	✓	0.761±0.0095	1.012±0.0088	51.3±0.65	0.6925±0.00019
✓	✗	0.49±0.012	0.9933±0.00019	74.5±0.49	0.579±0.0048
✗	✗	0.384±0.0018	1.0141±0.00066	0.49997±0.000039	0.698±0.0013

Table B.2: CNN-LSTM IMDB simulation accuracy ablation at $T = 1$: with *both* layerwise preconditioning and cyclical time stepping we can achieve high inference accuracy as measured by configurational and kinetic temperature diagnostics. Just using one (either preconditioning or cyclical time stepping) is insufficient for high inference accuracy. This is markedly different from the results obtained for ResNet-20 CIFAR-10 (Table B.1), indicating that perhaps the ResNet posterior is easier to sample from.

"dirty" to distinguish them from clean likelihood functions without such modifications. Our discussion will suggest that these techniques can be seen as a computational efficient "*Jensen posterior*" approximation of a proper Bayesian posterior of another model. Our analysis builds on and generalizes previous Bayesian interpretations, (Noh et al., 2017; Atanov et al., 2018; Shekhovtsov & Flach, 2018; Nalisnick et al., 2019; Inoue, 2019). In Section B.11.4 we perform an experiment to demonstrate that the dirty likelihood cannot explain cold posteriors.

B.11.1 Augmented Latent Model

To accommodate popular deep learning methods we first augment the probabilistic model $p(y|x, \boldsymbol{\theta})$ itself by adding a *latent variable* z. The augmented model is $p(y|x, z, \boldsymbol{\theta})$ and

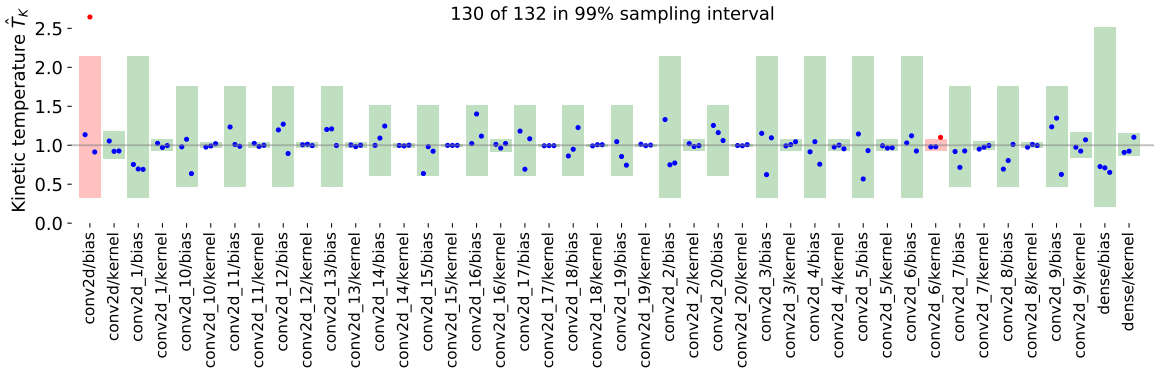


Figure B.8: ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **with preconditioning** and **with cosine time stepping schedule**. The green bars show the 99% true sampling distribution of the Kinetic temperature sample. The blue dots show the actual kinetic temperature samples at the end of sampling. About 1% of variables should be outside the green boxes, which matches the empirical count (2 out of 132 samples), indicating an accurate simulation of the Langevin dynamics at the end of each cycle.

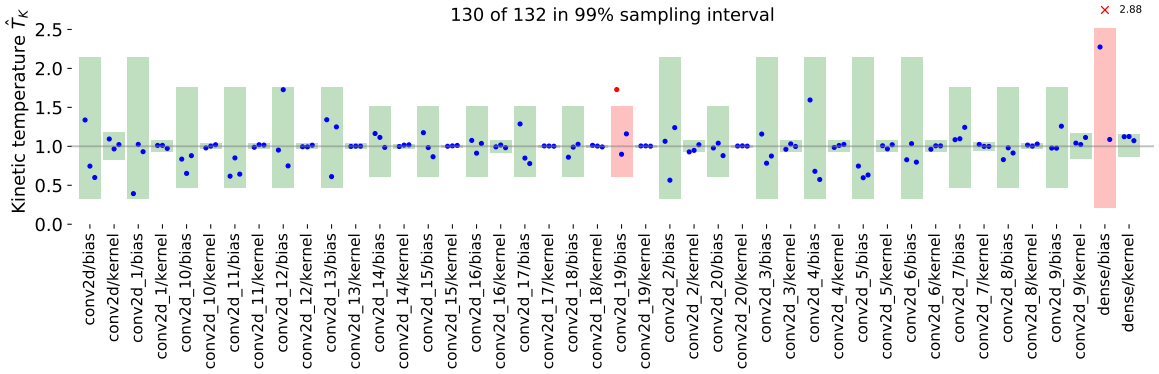


Figure B.9: ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **without preconditioning** but **with cosine time stepping schedule**. Two out of 132 variables are outside the 99% hpd region, indicating accurate simulation.

we can obtain the *effective model* $p(y|x,\boldsymbol{\theta}) = \int p(y|x,z,\boldsymbol{\theta})\, p(z) dz$. For a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1,\ldots,n}$, where we denote $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, the resulting model has as likelihood function in $\boldsymbol{\theta}$ that is the *marginal likelihood*, obtained by integrating over all z_i variables,

$$p(Y \,|\, X, \boldsymbol{\theta}) \;=\; \prod_{i=1}^{n} p(y_i \,|\, x_i, \boldsymbol{\theta}) \tag{B.36}$$

$$\;=\; \prod_{i=1}^{n} \mathbb{E}_{z_i \sim p(z_i)}[p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})]. \tag{B.37}$$

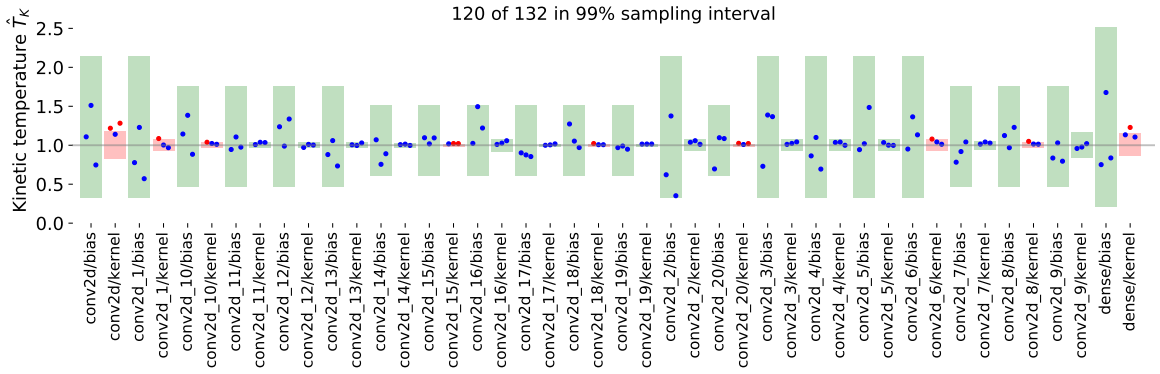


Figure B.10: ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **with preconditioning** but **without cosine time stepping schedule** (flat schedule). 12 out of 132 variables are too hot (boxes in red) and lie outside the acceptable region, indicating an inaccurate simulation of the Langevin dynamics. However, there is a marked improvement due to preconditioning compared to no preconditioning (Figure B.11).

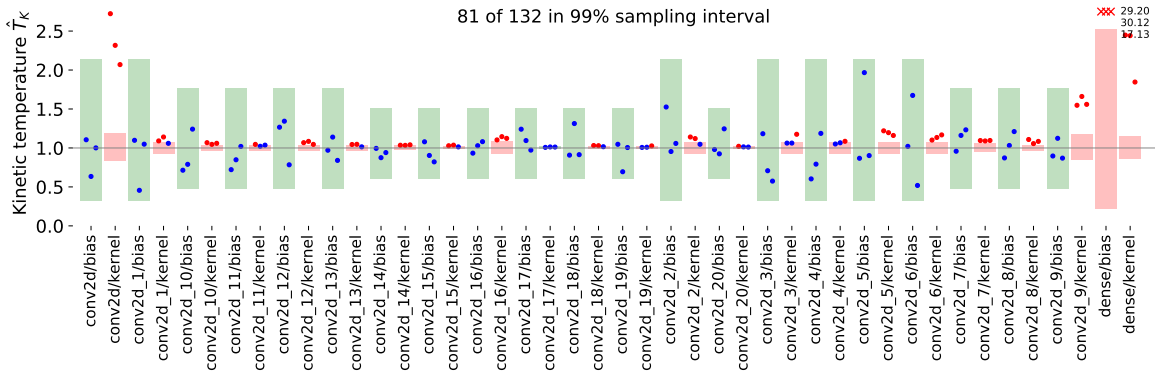


Figure B.11: ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **without preconditioning** and **without cosine time stepping schedule** (flat schedule). 51 out of 132 kinetic temperature samples are too hot (shaded in red) and lie outside the acceptable region, sometimes severely so, indicating a very poor simulation accuracy for the Langevin dynamics.

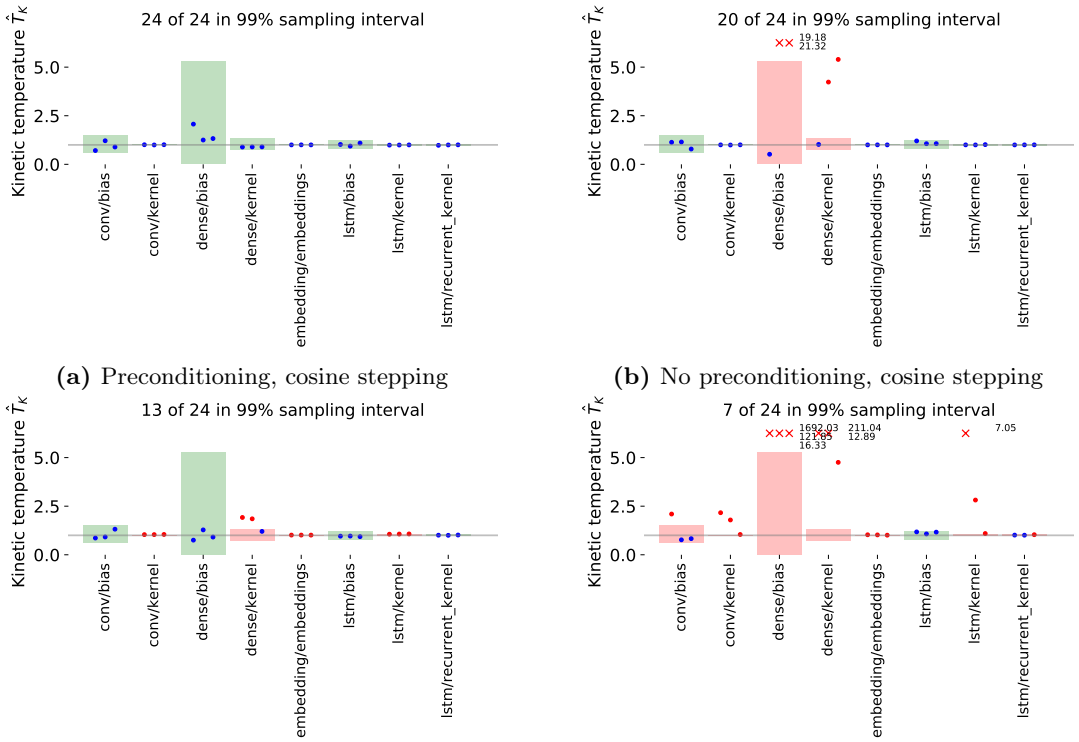
Note that in (B.37) the latent variable z_i is integrated out and therefore the marginal likelihood is a deterministic function.

B.11.2 Log-likelihood Bound and Jensen Posterior

Given a prior $p(\boldsymbol{\theta})$ the log-posterior for the augmented model in Figure B.13 takes the form

$$\log p(\boldsymbol{\theta} \,|\, \mathcal{D}) \tag{B.38}$$

$$= C + \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log \mathbb{E}_{z_i \sim p(z_i)}[p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})], \tag{B.39}$$



(a) Preconditioning, cosine stepping

(b) No preconditioning, cosine stepping

(c) Preconditioning, no cosine stepping

(d) No preconditioning, no cosine stepping

Figure B.12: CNN-LSTM IMDB Langevin per-variable kinetic temperature estimates at temperature $T = 1$ for four different simulation settings: with and without preconditioning, with and without cosine time stepping. The only accurate simulation is obtained with *both* preconditioning *and* cosine time stepping.

where we can now apply *Jensen's inequality*, $f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$ for concave $f = \log$,

$$\geq C + \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \mathbb{E}_{z_i \sim p(z_i)}[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})], \tag{B.40}$$

where $C = -\log p(Y|X)$ is the negative model evidence and is constant in $\boldsymbol{\theta}$. We call equation (B.40) the *Jensen bound* to the log-posterior $\log p(\boldsymbol{\theta}|\mathcal{D})$.

Jensen Posterior. Because we can estimate (B.40) in an unbiased manner, we will see that many popular methods such as dropout and data augmentation can be cast as special cases of the Jensen bound. We also define the *Jensen posterior* as the posterior distribution associated with (B.40). Formally, the Jensen posterior is

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) :\propto \tag{B.41}$$

$$p(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)}\left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right). \tag{B.42}$$

Given this object, can we relate its properties to the properties of the full posterior, and can the Jensen posterior serve as a meaningful surrogate to the true posterior? We first observe that $p_J(\boldsymbol{\theta} \,|\, \mathcal{D})$ indeed defines a probability distribution over parameters: with a proper prior $p(\boldsymbol{\theta})$, we have $p(\boldsymbol{\theta} \,|\, \mathcal{D}) \geq p_J(\boldsymbol{\theta} \,|\, \mathcal{D})$ by (B.39–B.40), thus $\int p_J(\boldsymbol{\theta} \,|\, \mathcal{D})\, \mathrm{d}\boldsymbol{\theta} \leq \int p(\boldsymbol{\theta} \,|\, \mathcal{D})\, \mathrm{d}\boldsymbol{\theta} < \infty$.

Jensen Prior. We now show that the Jensen posterior can be interpreted as a full Bayesian posterior in a different model. In particular, we give a construction which retains the likelihood of the original model but modifies the prior. In the function that re-weights the prior the data set appears; this is not to be understood as a prior which depends on the observed data. Instead, we can think of this as an existence proof, that is, if we were to have chosen this modified prior then the resulting Jensen posterior under the modified Jensen prior corresponds to the full Bayesian posterior under the original prior.

In a sense the result is vacuous because any desirable posterior can be obtained by such re-weighting. However, the proof illustrates the structure of how the Jensen posterior deviates from the true posterior through a set of weighting functions; each weighting function measures a local *Jensen gap* related to each instance. Although we did not pursue this line, the local Jensen gap (B.47) can be numerically estimated and may prove to be a useful quantity in itself.

Proposition 2 (Jensen Prior)**.** *For a proper prior $p(\boldsymbol{\theta})$ and a fixed dataset $\mathcal{D}$, we can define a prior $p_J(\boldsymbol{\theta})$ such that when using this modified prior in the Jensen posterior we have*

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) = p(\boldsymbol{\theta} \,|\, \mathcal{D}). \tag{B.43}$$

In particular, this implies that any Jensen posterior can be interpreted as the posterior distribution of the same model under a different prior.

Proof. We have the true posterior

$$p(\boldsymbol{\theta} \,|\, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{i=1}^{n} \int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i)\, \mathrm{d}z_i, \tag{B.44}$$

and the Jensen posterior as

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) := p(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), \tag{B.45}$$

respectively. If we define the *Jensen prior*,

$$p_J(\boldsymbol{\theta}) :\propto w(\boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{B.46}$$

where we set the weighting function $w(\boldsymbol{\theta}) := \prod_{i=1}^{n} w_i(\boldsymbol{\theta})$, with the individual weighting functions defined as

$$w_i(\boldsymbol{\theta}) := \frac{\int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i)\, \mathrm{d}z_i}{\exp\left(\mathbb{E}_{z_i \sim p(z_i)} [\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})]\right)}. \tag{B.47}$$

Due to Jensen's inequality we have $w_i(\boldsymbol{\theta}) \leq 1$ and hence $w(\boldsymbol{\theta}) \leq 1$ and thus $p_J(\boldsymbol{\theta})$ is normalizable. Using $p_J(\boldsymbol{\theta})$ as prior in (B.45) we obtain

$$\begin{aligned}
& p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) && \text{(B.48)}\\
& \propto p_J(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), && \text{(B.49)}\\
& = p(\boldsymbol{\theta}) \left(\prod_{i=1}^{n} w_i(\boldsymbol{\theta})\right) && \text{(B.50)}\\
& \qquad \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), && \text{(B.51)}\\
& = p(\boldsymbol{\theta}) \prod_{i=1}^{n} \int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i)\, \mathrm{d}z_i && \text{(B.52)}\\
& \propto p(\boldsymbol{\theta} \,|\, \mathcal{D}). && \text{(B.53)}
\end{aligned}$$

This constructively demonstrates the result (B.43). □

We now interpret current deep learning methods as optimizing the Jensen posterior.

B.11.3 Deep Learning Techniques Optimize Jensen Posteriors

Dropout. In *dropout* we sample random binary masks $z_i \sim p(z_i)$ and multiply network activations with such masks (Srivastava et al., 2014). Specializing the above latent variable

model to dropout gives an interpretation of doing maximum aposteriori (MAP) estimation on the Jensen posterior $p_J(\boldsymbol{\theta} \mid X, Y)$.

The connection between dropout and applying Jensen's bound has been discovered before by several groups Noh et al. (2017), Nalisnick et al. (2019), Inoue (2019), and contrasts sharply with the variational inference interpretation of dropout, (Kingma et al., 2015; Gal & Ghahramani, 2016). Recent variants of dropout such as *noise-in* (Dieng et al., 2018) can also be interpreted in the same way.

The Jensen prior interpretation justifies the use of standard dropout in Bayesian neural networks: the inferred posterior is the Jensen posterior which is also a Bayesian posterior under the Jensen prior.

Data Augmentation. Data augmentation is a simple and intuitive way to insert high-level prior knowledge into neural networks: by targeted augmentation of the available training data we can encode invariances with respect to natural transformation or noise, leading to better generalization, Perez & Wang (2017).

Data augmentation is also an instance of the above latent variable model, where z_i now corresponds to randomly sampled parameters of an augmentation, for example, whether to flip an image along the vertical axis or not.

Interestingly, the above model suggests that to obtain better predictive performance at test time, the posterior predictive should be obtained by averaging the individual posterior predictive distributions over multiple latent variable realizations. Indeed this is what early work on convolutional networks did, He et al. (2015b, 2016b), improving predictive performance significantly.

The Jensen prior interpretation again justifies the use of approximate Bayesian inference techniques targeting the Jensen posterior. In particular, our theory suggests that the dataset size n should *not* be adjusted to account for augmentation.

Batch Normalization. As a practical technique batch normalization (Ioffe & Szegedy, 2015) accelerates and stabilizes learning in deep neural networks. The model of Figure B.13 cannot directly serve to interpret batch normalization due to the dependence of batch normalization statistics on the batch. We therefore need to extend the model to incorporate a random choice of batches yielding continuous random batch normalization statistics as proposed earlier (Atanov et al., 2018; Shekhovtsov & Flach, 2018).

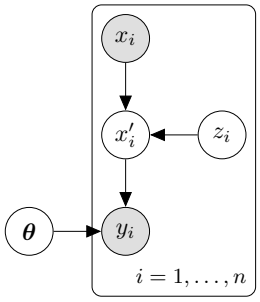


Figure B.13: Augmented model with added latent variable z_i.

Formally such variation of batch normalization corresponds to the model shown in Figure B.14, where $(x_i)_i \to \boldsymbol{\theta}$ signifies the additional randomness in $p(\boldsymbol{\theta}|X)$ due to random batches, and $(\boldsymbol{\theta}, x_i, z_i) \to x_i'$ are the resulting random outputs of the network, where z_i is a per-instance randomness source (Atanov et al., 2018).

With the above modifications all derivations in Section B.11.2 hold and batch normalization has a Jensen posterior. In particular, the Jensen interpretation also suggests to perform batch normalization at test-time, averaging over multiple different batches composed of training set samples.

B.11.4 Dirty Likelihood Experiment

The dirty likelihood hypothesis is plausible for the ResNet-20 experiments which use data augmentation and batch normalization, however, our CNN-LSTM model does have a clean likelihood function already.

To gain further confidence that this hypothesis cannot explain cold posterior we train a ResNet-20 without batch normalization or data augmentation.

Clean Likelihood ResNet Experiment: we disable data augmentation and replace batch normalization with filter response normalization, (Singh & Krishnan, 2019). Without data augmentation and without batch normalization we now have a clean likelihood function and SG-MCMC targets a true underlying Bayes posterior.

Figure B.15 on page 144 shows the predictive test performance as a function of temperature. We clearly see that for small temperatures $T \ll 1$ the removal of data augmentation and batch normalization leads to a higher standard error over the three runs, so that indeed data augmentation and batch normalization had a stabilizing effect on training and mitigated overfitting. However, for test accuracy the best performance by the SG-MCMC ensemble model is still achieved for $T < 1$. In particular, for test accuracy the best accuracy of $87.8 \pm 0.16\%$ is achieved at $T = 0.0193$, comparing to a worse predictive accuracy of $87.1 \pm 0.13\%$ at temperature $T = 1$. For test cross entropy the performance achieved at $T = 0.0193$ with 0.393 ± 0.015 is comparable to 0.3918 ± 0.0021 achieved at $T = 1$.

The clean likelihood ResNet experiment is slightly inconclusive as there is now a less marked improvement when going to lower temperatures. However, our CNN-LSTM IMDB model already had a clean likelihood function. Therefore, while dirty likelihoods may play a role in shaping the posterior that SG-MCMC methods simulate from they likely do not

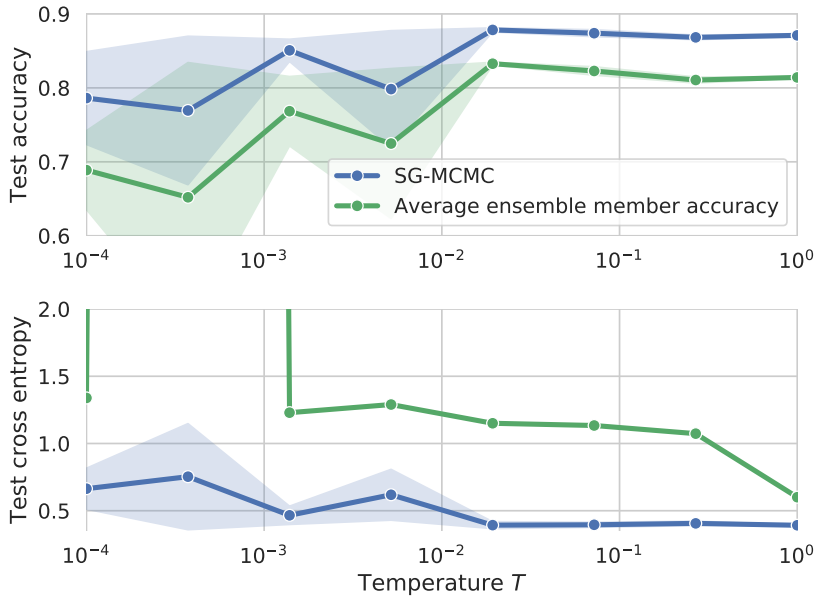


Figure B.15: ResNet-20 with filter response normalization (FRN) instead of batch normalization and without any use of data augmentation.

account for the cold posterior effect.

B.12 Prior Predictive Analysis for Different Prior Scales

Our experiments in Section 5.6.2 clearly demonstrate that the prior $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$ is bad in that it places prior mass on the same highly concentrated class probabilities for all training instances.

What other priors could we use? The literature contains significant prior work on this question. Neal (1995) examined priors for shallow neural networks and identified scaling laws and correspondence to Gaussian process kernels. Recently a number of works added to Neal's analysis by extending the results to deep and wide neural networks (Lee et al., 2018; de G. Matthews et al., 2018; Yang, 2019), convolutional networks (Garriga-Alonso et al., 2019), and Bayesian neural networks (Novak et al., 2019).

A related line of work explores random functions defined by the initialization process of a deep neural network. Glorot & Bengio (2010a) and He et al. (2015b) developed efficient random initialization schemes for deep neural networks and a more formal analysis of information flow in random functions defined by neural networks is given by Schoenholz et al. (2017) and Hayou et al. (2018). All these works derive variance-scaling laws for independent Gaussian priors. The precise scaling law depends on the network layer and the activation function being used. For the same architecture and activation the scaling laws generally agree with those obtained from the Gaussian process perspective. Figure 5.12 shows that the cold posterior effect is present regardless of the scaling of the variance of the Normal

prior. In the following we investigate certain scaling laws of the prior more detailed.

B.12.1 He-Scaled Normal Prior, $\mathcal{N}(0, I)$ for Biases

To remain as close as possible to our existing setup we investigate a He-scaling prior, equation (14) in (He et al., 2015b).

$$p(\boldsymbol{\theta}_j) = \mathcal{N}\left(0, \frac{2}{b_j}\right), \tag{B.54}$$

where b_j is the *fan-in* of the j'th layer.[5]

The scaling law derived by He et al. (2015b) does not cover the bias terms in a model. This is due to the work considering only initialization—(He et al., 2015b) initialized all biases to zero—whereas we would like to have proper priors for all model variables. We therefore choose the original $\mathcal{N}(0, I)$ prior for all bias variables in our model.

He-scaled Prior Predictive Experiment: For our ResNet-20 setup on CIFAR-10 we use our He-scaled-Normal prior to once again carry out the prior predictive experiment that was originally done in Section 5.6.2, Figures 5.7 and 5.8. Figure B.16 show the prior predictive results for the new prior. The basic conclusion remains unchanged: despite scaling the convolution weights and dense layer weights by the He-scaling law in the prior the prior predictive distributions remain highly concentrated around the same distribution for all training instances.

Why do functions under this prior remain concentrated? Perhaps it is due to the loose $\mathcal{N}(0, I)$ prior for the bias terms such that any concentration in early layers is amplified in later layers? We investigate this further in Section B.12.2.

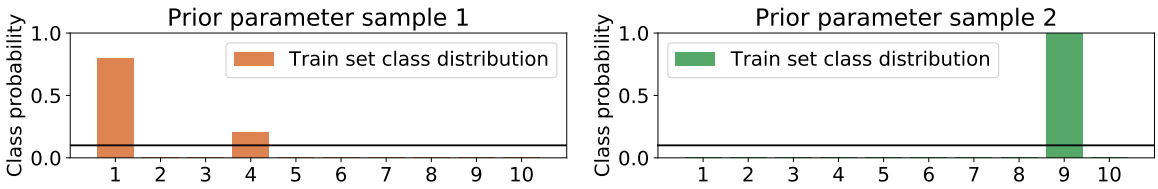
He-scaled Prior ResNet-20 CIFAR-10 Experiment: We also perform the original cold posterior experiment from Chapter 5 with the He-scaling Normal prior. We show the temperature-dependence curves for test accuracy and test cross-entropy in Figure B.17. The overall performance drops compared to the $\mathcal{N}(0, I)$ prior, but the cold posterior effect clearly remains. With this result and the result from the prior predictive study we can conclude that a simple Normal scaling correction is not enough to yield a sensible prior.

B.12.2 He-Scaled Normal Prior, $\mathcal{N}(0, \epsilon I)$ for Biases

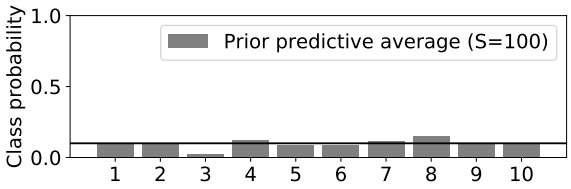
In this section we experiment with He-scaling and a very small scale for the bias prior. There are two motivations for such experimentation: *first*, He-scaling was originally proposed by He et al. (2015b) for initializing deep convolutional neural networks and in their initialization all bias terms were initialized to zero. *Second*, bias terms influence a large number of downstream activations and getting the scale wrong for our bias priors may have the large concentration effect that we observe in the previous prior predictive experiments.

We therefore propose to use a He-scaling Normal prior for all `Conv2D` and `Dense` layer weights and to use a $\mathcal{N}(0, \epsilon I)$ prior for all bias terms. Here we use $\epsilon = \sigma^2$ with $\sigma = 10^{-6}$,

[5] For a `Dense` layer the fan-in is the number of input dimensions, for a `Conv2D` layer with a kernel of size k-by-k and d input channels the fan-in is $b_j = k^2 d$.



(a) Typical predictive distributions for 10 classes under the prior, averaged over the entire training set, $\mathbb{E}_{x\sim p(x)}[p(y|x, \boldsymbol{\theta}^{(i)})]$. Each plot is for one sample $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta})$. Given a sample $\boldsymbol{\theta}^{(i)}$ the average training data class distribution is still highly concentrated around the same classes for all x.



(b) Prior predictive $\mathbb{E}_{x\sim p(x)}[\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta})}[p(y|x, \boldsymbol{\theta})]]$ over 10 classes for a Kaiming-scaling prior, estimated using $S = 100$ samples $\boldsymbol{\theta}^{(i)}$ and all training images.

Figure B.16: ResNet-20/CIFAR-10 prior predictive study for a He-scaled Normal prior for `Conv2D` and `Dense` layers and a $\mathcal{N}(0, I)$ prior for all bias terms. This prior concentrates prior mass on functions which output the *same* concentrated label distribution for *all* training instances. It is therefore a bad prior.

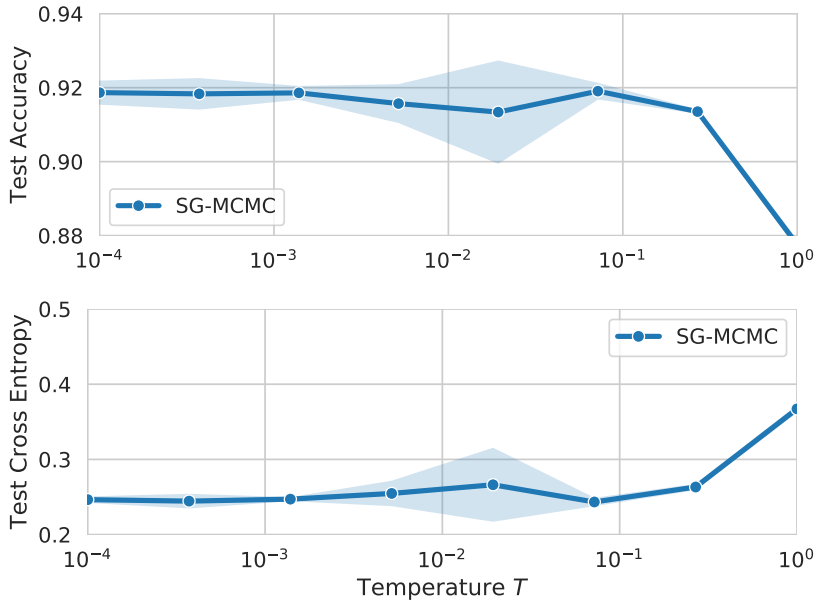
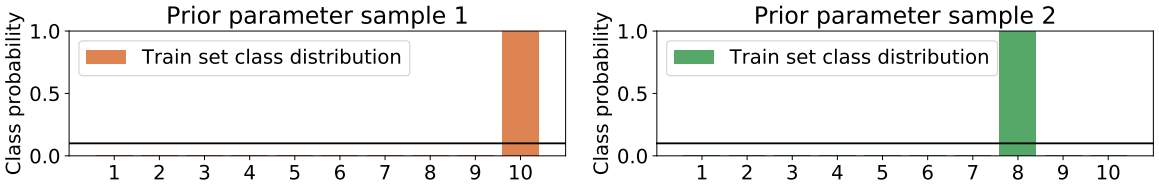
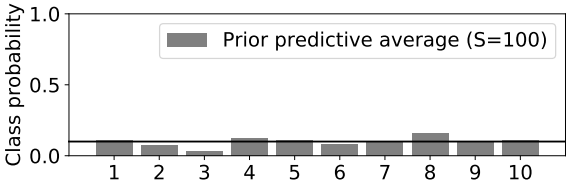


Figure B.17: ResNet-20 on CIFAR-10 with He-scaling Normal prior (He-scaled Normal for `Conv2D` and `Dense` layers, and $\mathcal{N}(0, I)$ for all bias terms). The cold posterior effect remains: the poor predictive performance of the Bayes posterior at $T = 1$ holds for both accuracy and cross-entropy.



(a) Typical predictive distributions for 10 classes under the prior, averaged over the entire training set, $\mathbb{E}_{x\sim p(x)}[p(y|x, \boldsymbol{\theta}^{(i)})]$. Each plot is for one sample $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta})$. Given a sample $\boldsymbol{\theta}^{(i)}$ the average training data class distribution is still highly concentrated around the same classes for all x despite using a small $\mathcal{N}(0, \epsilon I)$ prior for biases.



(b) Prior predictive $\mathbb{E}_{x\sim p(x)}[\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta})}[p(y|x, \boldsymbol{\theta})]]$ over 10 classes for a Kaiming-scaling prior with a $\mathcal{N}(0, \epsilon I)$ prior for bias terms. We estimate the marginal distribution using $S = 100$ samples $\boldsymbol{\theta}^{(i)}$ and all training images.

Figure B.18: ResNet-20/CIFAR-10 prior predictive study for a He-scaled Normal prior for `Conv2D` and `Dense` layers and a $\mathcal{N}(0, \epsilon I)$ prior for all bias terms. This prior still concentrates prior mass on functions which output the *same* concentrated label distribution for *all* training instances. It is, therefore, a bad prior.

essentially sampling all bias terms close to zero as in the original initialization due to (He et al., 2015b).

He-scaled Prior, $\mathcal{N}(0, \epsilon I)$ Bias Prior Experiment: We draw ResNet-20 models from the prior and evaluate the predicted class distributions on the entire CIFAR-10 training set. Figure B.18a shows two prior draws and the resulting class distributions marginalized over the entire training set. Figure B.18b shows a marginal prior predictive, marginalized over $S = 100$ prior draws and the entire training distribution of 50,000 images. The resulting marginal prior predictive approaches the uniform distribution. However, the He-scaled prior with $\mathcal{N}(0, \epsilon I)$ for bias terms remains a bad prior: random draws place prior mass on the same concentrated class distribution for all training instances.

B.13 Tempering the Observation Model?

In (Wilson & Izmailov, 2020), Equation (4) a proposal is made to use a different likelihood function of the form

$$p_T(y|x, \boldsymbol{\theta}) :\propto p(y|x, \boldsymbol{\theta})^{1/T}. \tag{B.55}$$

It is claimed that with this adjusted observation model the cold posterior is simply the ordinary Bayes posterior of the modified model. Indeed, if we are to plug the right-hand side of (B.55) directly into our posterior energy function (2.12), we obtain the cold posterior

energy function,

$$U_T(\boldsymbol{\theta}) := -\sum_{i=1}^{n} \frac{1}{T} \log p(y_i|x_i, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}). \tag{B.56}$$

The mistake in this derivation is to ignore that renormalization of $p_T(y|x, \boldsymbol{\theta})$ must be carried out because the normalizing constant is not invariant of $\boldsymbol{\theta}$. In particular, this is in contrast to typical applications of Bayes rule for posteriors, where we can indeed write $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ without worries, as here the normalizing constant does not depend on $\boldsymbol{\theta}$. One consequence of this mistake is that $U_T(\boldsymbol{\theta})$ is *not* necessarily the energy function of a Bayes posterior.

Instead, for the tempered observation model proposed by (Wilson & Izmailov, 2020) the correctly normalized observation likelihood is

$$p_T(y|x, \boldsymbol{\theta}) = \frac{p(y|x, \boldsymbol{\theta})^{1/T}}{\int p(y|x, \boldsymbol{\theta})^{1/T} \, \mathrm{d}y}. \tag{B.57}$$

Using this normalized observation model, the correct Bayes posterior energy corresponding to $p_T(y|x, \boldsymbol{\theta})$ is

$$\begin{aligned} \tilde{U}_T(\boldsymbol{\theta}) \quad := \quad & -\sum_{i=1}^{n} \frac{1}{T} \log p(y_i|x_i, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) \\ & + \sum_{i=1}^{n} \log \int p(y|x_i, \boldsymbol{\theta})^{1/T} \, \mathrm{d}y. \end{aligned} \tag{B.58}$$

Therefore, when the observation model is transformed as in (B.55) and as suggested by (Wilson & Izmailov, 2020), then in order to obtain a normalized observation model we must include the correction term (B.58) and this produces a modified energy function, $\tilde{U}_T(\boldsymbol{\theta})$, that differs from the actual cold posterior energy function $U_T(\boldsymbol{\theta})$.

Is there a way to "fix" this mistake? I.e. can one construct an observation model such that the resulting Bayes posterior corresponds to a tempered version of the Bayes posterior of the original observation model? For classification we found a way: we assign probability to a pseudo event "$\emptyset$" which cannot occur. To see this, assume a classification model $p(y|x)$ where $y \in \{1, 2, \ldots, K\}$. Clearly $\sum_{k=1}^{K} p(y = k|x) = 1$. Given a temperature $T \leq 1$ we define

$$\tilde{p}(y = k|x) := p(y = k|x)^{1/T}. \tag{B.59}$$

Clearly for $0 < T \leq 1$ we have that $f(x) = x^{1/T}$ is a monotonic function in $x \in [0, 1]$ and $f(x) = x^{1/T} \leq x$, and therefore $\sum_k \tilde{p}(y = k|x) \leq 1$. We absorb the remaining probability mass into a pseudo event "$\emptyset$",

$$\tilde{p}(y = \emptyset|x) := 1 - \sum_{k=1}^{K} \tilde{p}(y = k|x), \tag{B.60}$$

such that the resulting distribution $\tilde{p}$ over $K+1$ basic events sums to one,

$$\sum_{k\in\{1,2,\ldots,K,\emptyset\}} \tilde{p}(y=k|x) = 1. \tag{B.61}$$

Now observe that for any event in $\{1,2,\ldots,K\}$ that actually *can* occur we have

$$\log \tilde{p}(y=k|x) = \frac{1}{T}\log p(y=k|x), \tag{B.62}$$

that is, we have achieved the effect of temperature scaling when using $\tilde{p}$ as observation model. While formally possible, can we make sense of this transformation and introduction of a pseudo event?

To us it seems entirely non-Bayesian to artificially introduce events into a model while knowing with perfect certainty that these events cannot happen and then allow the model to assign probability mass to those events. It is non-Bayesian because our knowledge with respect to the new event is perfect: it cannot occur. Therefore a model should respect this knowledge of the world.

B.14 Details: Generation of a Synthetic Dataset Based on an MLP Drawn From its Prior Distribution

In this section, we describe how we generate a synthetic dataset based on a multi-layer perceptron (MLP) drawn from its prior distribution, as used in Section 5.5.2.

We generate synthetic data by (i) drawing a MLP from its prior distribution, i.e., $\texttt{mlp}_{\boldsymbol{\theta}}$ with $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$, (ii) sampling input data point x's $\in \mathbb{R}^5$ from a standard normal distribution and (iii) sampling label y's $\in \{1,2,3\}$ from the resulting logits $\texttt{mlp}_{\boldsymbol{\theta}}(x)$. We take $\texttt{mlp}_{\boldsymbol{\theta}}$ to be of depth 2, with 10 units and relu activation functions. We generate $n = 100$ points for inference and 10,000 for evaluations.

The choice of $p(\boldsymbol{\theta})$ requires some care. On the one hand, a naive choice of normal priors with unit standard deviation leads to a degenerated dataset that concentrates all its outputs on a single class. On the other hand, normal priors with a smaller standard deviation[6], e.g., 0.05, lead to a less spiky label distribution but with little dependence on the input x's.

As a result, we considered a He normal prior (He et al., 2015b) for the weights of $\texttt{mlp}_{\boldsymbol{\theta}}$ and a normal prior, with standard deviation 0.05, for the bias terms. We similarly adapted the choice of the priors for the MLPs used to learn over the data generated in this way.

B.15 Details about Hamiltonian Monte Carlo

In this section, we describe practical considerations about Hamiltonian Monte Carlo (HMC) and present further results about its comparison with SG-MCMC (see Section 5.5.2).

HMC mainly exposes four hyperparameters that need to be set (Neal et al., 2011):

[6]Default value of `tf.random_normal_initializer`.

- The number L of steps of the leapfrog integrator,
- The step size ε in the leapfrog integrator,
- The number b of steps of the burn-in phase,
- The number S of samples to generate.

B.15.1 Hyperparameter choices

In our experiments with HMC, we have set $S = 2500$, generating a total of 25000 samples after the burn-in phase and keeping one sample every ten samples.

For the burn-in phase, we investigated in preliminary experiments the effect of varying the number of steps $b \in \{500, 1000, 5000\}$, noticing that our diagnostics (as later described) started to stabilize for $b = 1000$, so that we decided to use $b = 5000$ out of precaution (even though it may not be the most efficient option).

We thereafter searched a good combination of leapfrog steps and step size for $L \in \{5, 10, 100\}$ and $\varepsilon \in \{0.001, 0.01, 0.1\}$.

B.15.2 Convergence monitoring

We monitor convergence by first inspecting trace plots and second by computing standard diagnostics, namely the effective sample size (ESS) (Brooks et al., 2011) and the potential scale reduction (PSRF) (Gelman & Rubin, 1992).

Trace plots. In Figures B.20-B.21-B.22, we detail the inspection of the 5 different chains for the choice $L = 100$ and $\varepsilon = 0.1$ (which corresponds to the results of the sampler shown in Chapter 5). As practical diagnostic tools, we consider *trace plots* where we monitor the evolution of some statistics with respect to the generated HMC samples (e.g., see Section 24.4 in Murphy (2012), and references therein, for an introduction in a machine learning context). We compute trace plots for different depths of the MLP (in $\{1, 2, 3\}$) and different[7] temperatures, $T \in \{0.001, 0.0024, 0.014, 1.0\}$.

In addition to monitoring the evolution of the cross entropy for $S' \in \{1, 2, \ldots, S\}$ HMC samples (see Figure B.20), we also consider the following statistics:

- **Mean of the predictive entropy:** Let us denote by $\mathcal{D}_{\text{held-out}}$ the held-out set of pairs (x, y) and $\mathcal{E}_{\boldsymbol{\theta}}(x)$ the entropy of the softmax output at the input x

$$\mathcal{E}_{\boldsymbol{\theta}}(x) = -\sum_{c} p(y = c|x, \boldsymbol{\theta}) \log p(y = c|x, \boldsymbol{\theta}),$$

  together with its average over the held-out set

$$\mathcal{E}_{\boldsymbol{\theta}} = \frac{1}{|\mathcal{D}_{\text{held-out}}|} \sum_{(x,y)\in\mathcal{D}_{\text{held-out}}} \mathcal{E}_{\boldsymbol{\theta}}(x).$$

[7] We limit ourselves to four temperatures to avoid clutter.

For $S' \in \{1, 2, \ldots, S\}$ samples collected along the trajectory of HMC, we report in Figure B.21 the estimate

$$\hat{\mathcal{E}} = \frac{1}{S'} \sum_{s=1}^{S'} \mathcal{E}_{\boldsymbol{\theta}_s} \approx \bar{\mathcal{E}} = \int \mathcal{E}_{\boldsymbol{\theta}} \cdot p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta},$$

which we refer to as the mean of the predictive entropy.

- **Standard deviation of the predictive entropy:** We also consider the monitoring of the second moment of the predictive entropy. With the above notation, we estimate

$$\frac{1}{S'-1} \sum_{s=1}^{S'} (\mathcal{E}_{\boldsymbol{\theta}_s} - \hat{\mathcal{E}})^2 \approx \int (\mathcal{E}_{\boldsymbol{\theta}} - \bar{\mathcal{E}})^2 \cdot p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}$$

and report its square root in Figure B.22, which we refer to as the standard deviation of the predictive entropy.

As a general observation, we can see on Figures B.20-B.21-B.22 that, overall, the 5 different chains tend to exhibit a converging behavior for the three examined statistics, with typically more dispersion as the depth and the temperature increase (which is reflected by the ranges of the y-axis in the plots of Figures B.20-B.21-B.22 that get wider as T and the depth become larger).

ESS and PSRF. The effective sample size (ESS) (Brooks et al., 2011) measures how independent the samples are in terms of the auto-correlations within the sequence at different lags. The potential scale reduction factor (PSRF) Gelman & Rubin (1992) assesses the convergence of the chains (to the same target distribution) by testing for equality of means.

We computed ESS and PSRF for our HMC simulation (with 100 leapfrog steps and a step size of 0.1, as reported in Chapter 5). We used the TFP implementations `tfp.mcmc.{effective_sample_size, potential_scale_reduction}`. Figure B.19 (left, middle) displays the ESS and PSRF with respect to the different temperature levels, for the 3 MLP depths. Both ESS and PSRF were averaged over the model parameters.

We observe that in the regime T in $[0.05, 1]$, the diagnostics indicate an approximate convergence (PSRF < 1.05 and ESS in $[1800, S]$, with $S = 2500$ total samples) for the 3 MLP depths. On the other hand, in the regime T in $[0.001, 0.05]$, the diagnostics only continue to indicate an approximate convergence for the depth 1. For depths 2 and 3, both diagnostics substantially degrade, e.g., ESS down to ≈ 189 for depth 3.

B.15.3 KL divergence between predictive distributions

In Section 5.5.2, we compare side by side the cross-entropy of SG-MCMC and HMC for the different temperature levels, exhibiting a close agreement.

As an alternative visualization of this comparison, we computed the (symmetrized) KL divergence between the SG-MCMC and HMC predictive distributions (i.e., in our setting, categorical distributions with 3 classes).

For SG-MCMC and HMC (instantiated with 100 leapfrog steps and a step size of 0.1, as reported in Chapter 5), Figure B.19 (right) displays the (symmetrized) KL with respect to the different temperature levels, for the 3 MLP depths (averaged over the seeds). We observe that all KLs are small (in the order of $\approx 10^{-5}$ for depth 1, and $\approx 10^{-3}$ for depths 2 and 3).

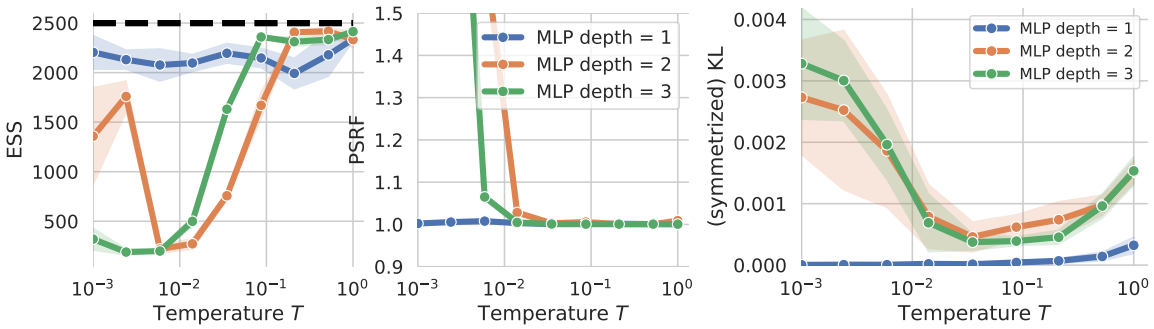


Figure B.19: For HMC (instantiated with 100 leapfrog steps and a step size of 0.1, as reported in Section 5.5.2), we report the effective sample size (left) and potential scale reduction factor (middle) with respect to the different temperature levels. On the left plot, the black dash line corresponds to the $S = 2500$ samples and ESS=S indicates no correlation in the sequences. For PSRF, approximate convergence is generally considered when PSRF < 1.2 (Gelman & Rubin, 1992). (right) KL divergence between the predictive distributions of HMC and SG-MCMC.

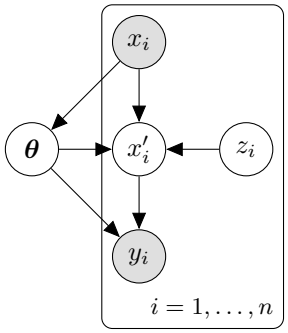


Figure B.14: Augmented model for batch normalization.

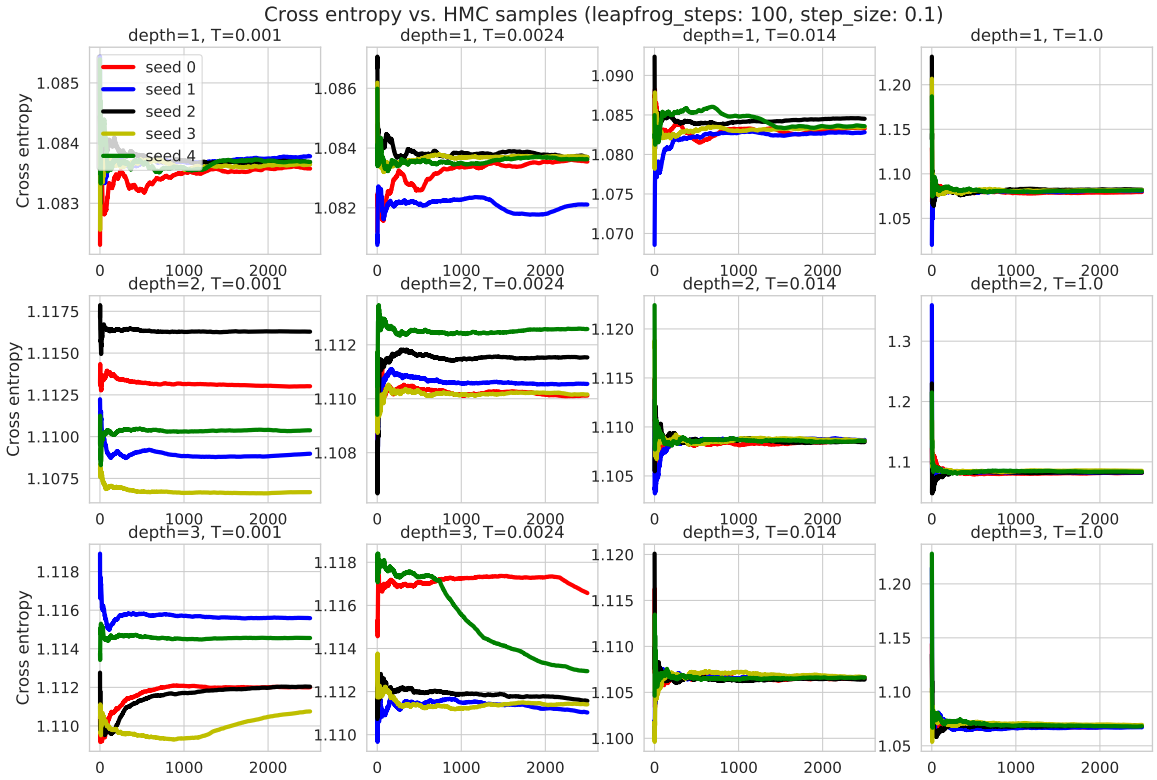


Figure B.20: Trace plots of the cross entropy: We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). Overall, the chains exhibit a converging behavior, with typically more dispersion as the depth and the temperature increase (which is reflected by the ranges of the y-axis that get wider as T and the depth increase).

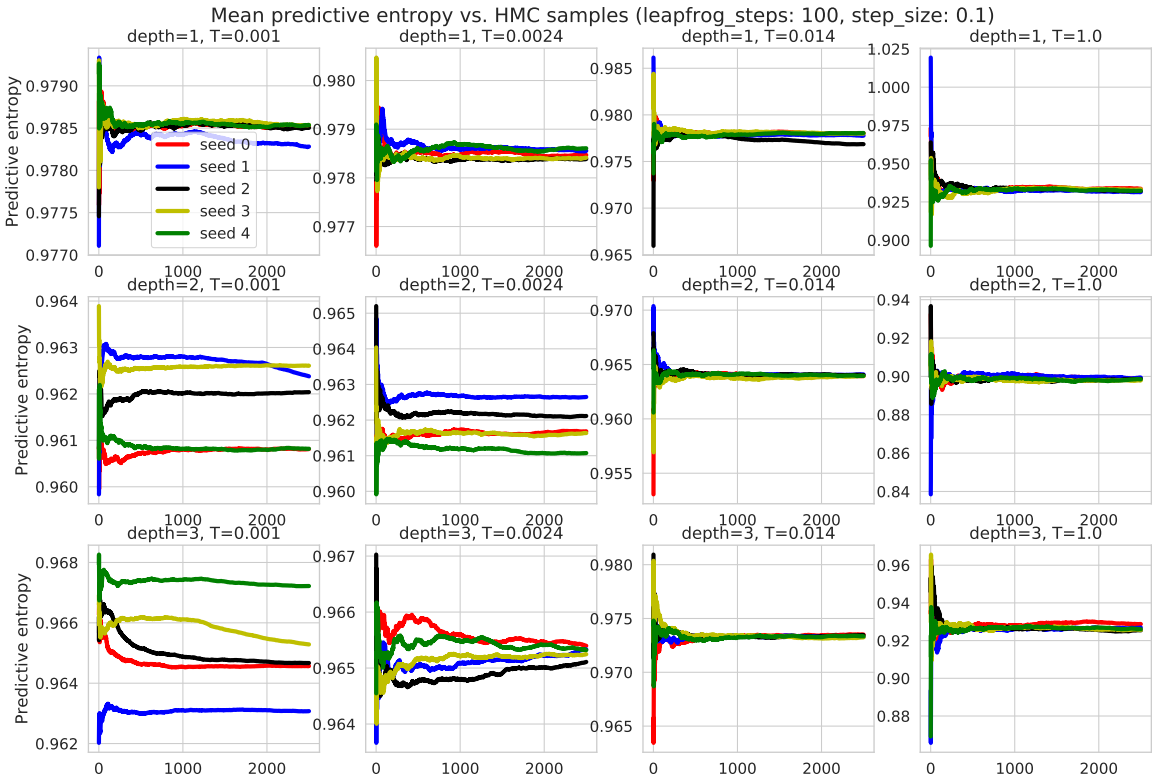


Figure B.21: Trace plots of the mean predictive entropy (see definition in Section B.15). We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). See further discussions in Figure B.20.

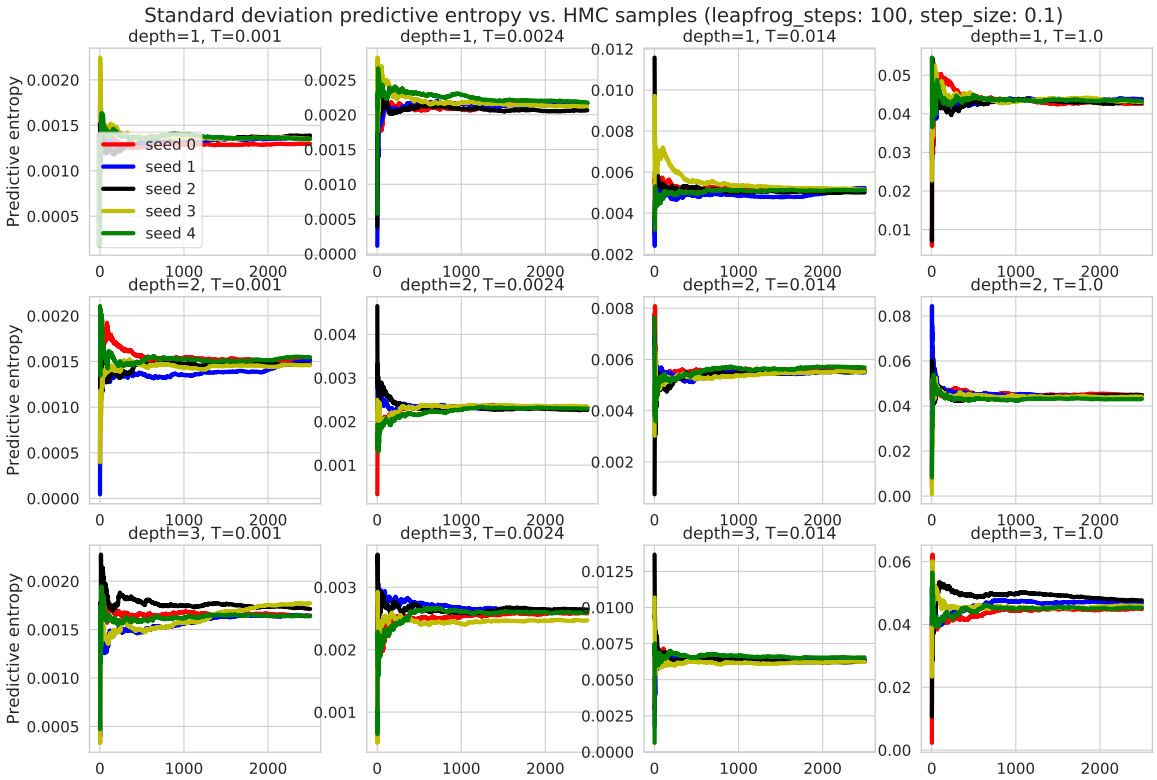


Figure B.22: Trace plots of the standard deviation of the predictive entropy (see definition in Section B.15). We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). See further discussions in Figure B.20.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

Allen, J. Short term spectral analysis, synthesis, and modification by discrete fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(3):235–238, 1977.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *Eigth International Conference on Learning Representations (ICLR 2020)*, 2020.

Atanov, A., Ashukha, A., Molchanov, D., Neklyudov, K., and Vetrov, D. Uncertainty estimation via stochastic batch normalization. *arXiv preprint arXiv:1802.04893*, 2018.

Babiański, M., Pokora, K., Shah, R., Sienkiewicz, R., Korzekwa, D., and Klimkov, V. On granularity of prosodic representations in expressive text-to-speech. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pp. 892–899, 2023. doi: 10.1109/SLT54892.2023.10022793.

Bae, J., Zhang, G., and Grosse, R. Eigenvalue corrected noisy natural gradient. *arXiv preprint arXiv:1811.12565*, 2018.

Baldock, R. J. and Marzari, N. Bayesian neural networks at finite temperature. *arXiv preprint, arXiv:1904.04154*, 2019.

Barber, D. and Bishop, C. M. Ensemble learning for multi-layer networks. In *Advances in neural information processing systems*, pp. 395–401, 1998a.

Barber, D. and Bishop, C. M. Ensemble learning for multi-layer networks. In *Advances in neural information processing systems*, pp. 395–401, 1998b.

Berger, J. O. *Statistical decision theory and Bayesian analysis*. Springer, 1985.

Berger, J. O. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

Betancourt, M. and Girolami, M. Hamiltonian Monte Carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, 79:30, 2015.

Bhattacharya, A., Pati, D., Yang, Y., et al. Bayesian fractional posteriors. *The Annals of Statistics*, 47(1):39–66, 2019.

Bińkowski, M., Donahue, J., Dieleman, S., Clark, A., Elsen, E., Casagrande, N., Cobo, L. C., and Simonyan, K. High fidelity speech synthesis with adversarial networks. *International Conference on Learning Representations*, 2020.

Blackman, R. B. and Tukey, J. W. The measurement of power spectra from the point of view of communications engineering—part i. *Bell System Technical Journal*, 37(1):185–282, 1958.

Blei, D., Ranganath, R., and Mohamed, S. Variational inference: Foundations and modern methods. *NIPS Tutorial*, 2016.

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

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015a.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 37:1613–1622, 2015b.

Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023a.

Borsos, Z., Sharifi, M., Vincent, D., Kharitonov, E., Zeghidour, N., and Tagliasacchi, M. Soundstorm: Efficient parallel audio generation. *arXiv preprint arXiv:2305.09636*, 2023b.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Box, G. E. and Tiao, G. C. *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

Bracewell, R. N. and Bracewell, R. N. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

Brannon, W., Virkar, Y., and Thompson, B. Dubbing in practice: A large scale study of human localization with insights for automatic dubbing. *Transactions of the Association for Computational Linguistics*, 2021.

Brier, G. W. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Brooks, S., Gelman, A., Jones, G., and Meng, X. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, 2011. ISBN 9781420079425. URL `https://books.google.de/books?id=qfRsAIKZ4rIC`.

Casanova, E., Weber, J., Shulby, C. D., Junior, A. C., Gölge, E., and Ponti, M. A. YourTTS: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*, pp. 2709–2720. PMLR, 2022.

Challis, E. and Barber, D. Concave gaussian variational approximations for inference in large-scale bayesian linear models. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 199–207, 2011.

Chen, C., Ding, N., and Carin, L. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pp. 2278–2286, 2015.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

Chen, T., Fox, E., and Guestrin, C. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pp. 1683–1691, 2014.

Cho, H., Jung, W., Lee, J., and Woo, S. H. SANE-TTS: Stable And Natural End-to-End Multilingual Text-to-Speech. In *Proc. Interspeech*, pp. 1–5, 2022. doi: 10.21437/Interspeech.2022-46.

Chollet, F. et al. Keras, 2015.

Conneau, A., Baevski, A., Collobert, R., Mohamed, A., and Auli, M. Unsupervised cross-lingual representation learning for speech recognition. In *Interspeech*, 2022.

de G. Matthews, A. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. In *ICLR*, 2018.

Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

Dieng, A. B., Ranganath, R., Altosaar, J., and Blei, D. M. Noisin: Unbiased regularization for recurrent neural networks. *arXiv preprint arXiv:1805.01500*, 2018.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., and Saurous, R. A. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. Bayesian sampling using stochastic gradient thermostats. In *Advances in neural information processing systems*, pp. 3203–3211, 2014.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

Earl, D. J. and Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

Effendi, J., Virkar, Y., Barra-Chicote, R., and Federico, M. Duration modeling of neural tts for automatic dubbing. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8037–8041, 2022a. doi: 10.1109/ICASSP43922.2022.9747158.

Effendi, J., Virkar, Y., Barra-Chicote, R., and Federico, M. Duration modeling of neural tts for automatic dubbing. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8037–8041. IEEE, 2022b.

Elias, I., Zen, H., Shen, J., Zhang, Y., Jia, Y., Weiss, R. J., and Wu, Y. Parallel tacotron: Non-autoregressive and controllable tts. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5709–5713. IEEE, 2021.

Farquhar, S., Osborne, M., and Gal, Y. Radial Bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning. *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, 2020a.

Farquhar, S., Smith, L., and Gal, Y. Try Depth instead of weight correlations: Mean-field is a less restrictive assumption for variational inference in deep networks. *Bayesian Deep Learning Workshop at NeurIPS*, 2020b.

Federico, M., Enyedi, R., Barra-Chicote, R., Giri, R., Isik, U., Krishnaswamy, A., and Sawaf, H. From speech-to-speech translation to automatic dubbing. In *IWSLT 2020*, 2020.

Flam-Shepherd, D., Requeima, J., and Duvenaud, D. Mapping gaussian process priors to bayesian neural networks. In *NIPS Bayesian deep learning workshop*, 2017.

Fushiki, T. et al. Bootstrap prediction and Bayesian prediction under misspecified models. *Bernoulli*, 11(4):747–758, 2005.

Gal, Y. *Uncertanity in Deep Learning*. University of Cambridge, 2016.

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

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. Deep convolutional networks as shallow gaussian processes. In *ICLR*, 2019.

Geisser, S. *An Introduction to Predictive Inference*. Chapman and Hall, New York, 1993.

Gelfand, A. E. and Smith, A. F. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

Gelman, A. and Rubin, D. B. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. Pac-bayesian theory meets bayesian inference. In *Advances in Neural Information Processing Systems*, pp. 1884–1892, 2016.

Gershman, S. and Goodman, N. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

Giordano, R., Broderick, T., and Jordan, M. I. Covariances, robustness and variational bayes. *The Journal of Machine Learning Research*, 19(1):1981–2029, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010a.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010b.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

Graves, A. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pp. 2348–2356, 2011a.

Graves, A. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pp. 2348–2356, 2011b.

Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. Ieee, 2013.

Griffin, D. and Lim, J. Signal estimation from modified short-time fourier transform. *IEEE Transactions on acoustics, speech, and signal processing*, 32(2):236–243, 1984.

Grünwald, P., Van Ommen, T., et al. Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it. *Bayesian Analysis*, 12(4):1069–1103, 2017.

Guo, Y., Du, C., Chen, X., and Yu, K. Emodiff: Intensity controllable emotional text-to-speech with soft-label guidance. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

Gupta, A. K. and Nagar, D. K. *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

Hafner, D., Tran, D., Lillicrap, T., Irpan, A., and Davidson, J. Noise contrastive priors for functional uncertainty. *arXiv preprint arXiv:1807.09289*, 2018.

Häggström, O. and Rosenthal, J. On variance conditions for markov chain clts. *Electronic Communications in Probability*, 12:454–464, 2007.

Hayou, S., Doucet, A., and Rousseau, J. On the selection of initialization and activation function for deep neural networks. *arXiv preprint arXiv:1805.08266*, 2018.

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

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016b.

Heber, F., Trstanova, Z., and Leimkuhler, B. Tati-thermodynamic analytics toolkit: Tensorflow-based software for posterior sampling in machine learning applications. *arXiv preprint arXiv:1903.08640*, 2019.

Heek, J. and Kalchbrenner, N. Bayesian inference for large scale image classification. *arXiv preprint arXiv:1908.03491*, 2019.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Hinton, G. and Van Camp, D. Keeping neural networks simple by minimizing the description length of the weights. In *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. Citeseer, 1993a.

Hinton, G. and Van Camp, D. Keeping neural networks simple by minimizing the description length of the weights. In *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*, 1993b.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Hoffman, M. D. and Gelman, A. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

Hsu, W.-N., Zhang, Y., Weiss, R. J., Zen, H., Wu, Y., Wang, Y., Cao, Y., Jia, Y., Chen, Z., Shen, J., et al. Hierarchical generative modeling for controllable speech synthesis. In *International Conference on Learning Representations*, 2019.

Hunt, A. J. and Black, A. W. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings*, volume 1, pp. 373–376. IEEE, 1996.

Inoue, H. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Isik, U., Giri, R., Phansalkar, N., Valin, J.-M., Helwani, K., and Krishnaswamy, A. PoCoNet: Better speech enhancement with frequency-positional embeddings, semi-supervised conversational data, and biased loss. In *Interspeech*, 2020.

Jaakkola, T. and Jordan, M. A variational approach to bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, volume 82, pp. 4, 1997.

Jansen, L. Robust Bayesian inference under model misspecification, 2013. Master thesis.

Jaynes, E. T. *Probability theory: The logic of science*. Cambridge university press, 2003.

Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Nguyen, P., Pang, R., Lopez Moreno, I., Wu, Y., et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, 31, 2018.

Jia, Y., Ramanovich, M. T., Wang, Q., and Zen, H. CVSS corpus and massively multilingual speech-to-speech translation. In *Proceedings of Language Resources and Evaluation Conference (LREC)*, pp. 6691–6703, 2022.

Jones, G. L. et al. On the Markov chain central limit theorem. *Probability surveys*, 1(299-320): 5–1, 2004.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

Karaletsos, T., Dayan, P., and Ghahramani, Z. Probabilistic meta-representations of neural networks. *arXiv preprint arXiv:1810.00555*, 2018.

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable bayesian deep learning by weight-perturbation in adam. *arXiv preprint arXiv:1806.04854*, 2018.

Kim, J., Kim, S., Kong, J., and Yoon, S. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33: 8067–8077, 2020.

Kim, J., Kong, J., and Son, J. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, pp. 5530–5540. PMLR, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Komaki, F. On asymptotic properties of predictive distributions. *Biometrika*, 83(2):299–313, 1996.

Kong, J., Kim, J., and Bae, J. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33: 17022–17033, 2020a.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020b.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009a.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009b.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Kullback, S. and Leibler, R. A. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Kumar, A., Sattigeri, P., and Balakrishnan, A. Variational inference of disentangled latent concepts from unlabeled observations. *International Conference on Learning Representations*, 2018.

Kumar, K., Kumar, R., De Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., De Brebisson, A., Bengio, Y., and Courville, A. C. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32, 2019.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30*. 2017.

Langevin, P. Sur la théorie du mouvement brownien. *Compt. Rendus*, 146:530–533, 1908.

Lao, J., Suter, C., Langmore, I., Chimisov, C., Saxena, A., Sountsov, P., Moore, D., Saurous, R. A., Hoffman, M. D., and Dillon, J. V. tfp.mcmc: Modern Markov chain Monte Carlo tools built for modern hardware, 2020.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, A., Chen, P.-J., Wang, C., Gu, J., Popuri, S., Ma, X., Polyak, A., Adi, Y., He, Q., Tang, Y., et al. Direct speech-to-speech translation with discrete units. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3327–3339, 2022.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. In *ICLR*, 2018.

Lee, S.-g., Ping, W., Ginsburg, B., Catanzaro, B., and Yoon, S. BigVGAN: A Universal Neural Vocoder with Large-Scale Training. *International Conference on Learning Representations*, 2023.

Lee, Y. and Kim, T. Robust and fine-grained prosody control of end-to-end speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5911–5915. IEEE, 2019.

Leimkuhler, B. and Matthews, C. *Molecular Dynamics*. Springer, 2016.

Leimkuhler, B., Matthews, C., and Vlaar, T. Partitioned integrators for thermodynamic parameterization of neural networks. *arXiv preprint arXiv:1908.11843*, 2019.

Li, C., Chen, C., Carlson, D., and Carin, L. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Liao, J. and Berg, A. Sharpening jensen's inequality. *The American Statistician*, 73(3): 278–281, 2019.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pp. 4114–4124. PMLR, 2019.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Louizos, C. and Welling, M. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pp. 1708–1716, 2016.

Louizos, C. and Welling, M. Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2218–2227. JMLR. org, 2017.

Luong, H.-T., Takaki, S., Henter, G. E., and Yamagishi, J. Adapting and controlling dnn-based speech synthesis using input codes. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4905–4909. IEEE, 2017.

Ma, Y.-A., Chen, T., and Fox, E. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, pp. 2917–2925, 2015.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pp. 142–150. Association for Computational Linguistics, 2011.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

MacKay, D. J. et al. Ensemble learning and evidence maximization. In *Proc. Nips*, volume 10, pp. 4083. Citeseer, 1995.

Mandt, S., McInerney, J., Abrol, F., Ranganath, R., and Blei, D. M. Variational tempering. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS*, JMLR Workshop and Conference Proceedings, 2016.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate Bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

Marlin, B. M., Khan, M. E., and Murphy, K. P. Piecewise bounds for estimating bernoulli-logistic latent gaussian models. In *Proceedings of the International Conference on Machine Learning*, pp. 633–640, 2011.

Masegosa, A. R. Learning under model misspecification: Applications to variational and ensemble methods. *arXiv preprint, arXiv:19012.08335*, 2019.

Masters, D. and Luschi, C. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

Matoušek, J. and Vít, J. Improving automatic dubbing with subtitle timing optimisation using video cut detection. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2385–2388. IEEE, 2012.

McGrayne, S. B. *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, & Emerged Triumphant from Two Centuries of C.* Yale University Press, 2011.

Miller, A. C., Foti, N. J., and Adams, R. P. Variational boosting: Iteratively refining posterior approximations. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2420–2429. JMLR. org, 2017.

Mishkin, A., Kunstner, F., Nielsen, D., Schmidt, M., and Khan, M. E. Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pp. 6245–6255, 2018.

Mitsui, K., Zhao, T., Sawada, K., Hono, Y., Nankaku, Y., and Tokuda, K. End-to-End Text-to-Speech based on latent representation of speaking styles using spontaneous dialogue. In *Proc. Interspeech*, pp. 2328–2332, 2022.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Nalisnick, E., Hernandez-Lobato, J. M., and Smyth, P. Dropout as a structured shrinkage prior. In *International Conference on Machine Learning*, pp. 4712–4722, 2019.

Neal, R. M. Bayesian learning via stochastic dynamics. In *Advances in neural information processing systems*, pp. 475–482, 1993.

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

Neal, R. M. et al. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

Nemenman, I., Shafee, F., and Bialek, W. Entropy and inference, revisited. In *Advances in neural information processing systems*, pp. 471–478, 2002.

Noh, H., You, T., Mun, J., and Han, B. Regularizing deep neural networks by noise: Its interpretation and optimization. In *Advances in Neural Information Processing Systems*, pp. 5109–5118, 2017.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Bayesian deep convolutional networks with many channels are gaussian processes. In *ICLR*, 2019.

Nowozin, S. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *Sixth International Conference on Learning Representations (ICLR 2018)*, 2018.

Ong, V. M.-H., Nott, D. J., and Smith, M. S. Gaussian variational approximation with a factor covariance structure. *Journal of Computational and Graphical Statistics*, 27(3): 465–478, 2018.

Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pp. 3918–3926. PMLR, 2018.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., and Khan, M. E. Practical deep learning with Bayesian principles. *arXiv preprint arXiv:1906.02506*, 2019a.

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., and Khan, M. E. Practical deep learning with bayesian principles. *arXiv preprint arXiv:1906.02506*, 2019b.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019a.

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

Pearce, T., Zaki, M., Brintrup, A., and Neely, A. Expressive priors in bayesian neural networks: Kernel combinations and periodic functions. *arXiv preprint arXiv:1905.06076*, 2019.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

Perez, L. and Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Peterson, C. A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019, 1987.

Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., and Kudinov, M. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pp. 8599–8608. PMLR, 2021.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

Prenger, R., Valle, R., and Catanzaro, B. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621. IEEE, 2019.

Rabiner, L. and Schafer, R. *Theory and applications of digital speech processing*. Prentice Hall Press, 2010.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

Raftery, A. E., Gneiting, T., Balabdaoui, F., and Polakowski, M. Using bayesian model averaging to calibrate forecast ensembles. *Monthly weather review*, 133(5):1155–1174, 2005.

Ramamoorthi, R. V., Sriram, K., and Martin, R. On posterior concentration in misspecified models. *Bayesian Anal.*, 10(4):759–789, 12 2015. doi: 10.1214/15-BA941.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Ranganath, R., Tran, D., and Blei, D. Hierarchical variational models. In *International Conference on Machine Learning*, pp. 324–333, 2016.

Rattcliffe, D., Wang, Y., Mansbridge, A., Karanasou, P., Moinet, A., and Cotescu, M. Cross-lingual Style Transfer with Conditional Prior VAE and Style Loss. In *Proc. Interspeech 2022*, pp. 4586–4590, 2022a. doi: 10.21437/Interspeech.2022-10572.

Rattcliffe, D., Wang, Y., Mansbridge, A., Karanasou, P., Moinet, A., and Cotescu, M. Cross-lingual Style Transfer with Conditional Prior VAE and Style Loss. In *Proc. Interspeech 2022*, pp. 4586–4590, 2022b. doi: 10.21437/Interspeech.2022-10572.

Recommendation, I. Bs. 1534-3: Method for the subjective assessment of intermediate quality levels of coding systems. *Geneva: International Telecommunications Union*, 2015.

Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32, 2019.

Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. Fastspeech 2: Fast and high-quality end-to-end text to speech. In *International Conference on Learning Representations*, 2020.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Robert, C. P. et al. *The Bayesian choice: from decision-theoretic foundations to computational implementation*, volume 2. Springer, 2007.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Saeki, T., Tachibana, K., and Yamamoto, R. DRSpeech: Degradation-Robust Text-to-Speech Synthesis with Frame-Level and Utterance-Level Acoustic Representation Learning. In *Proc. Interspeech*, pp. 793–797, 2022. doi: 10.21437/Interspeech.2022-294. URL `https://doi.org/10.21437/Interspeech.2022-294`.

Salimans, T., Knowles, D. A., et al. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

Särkkä, S. and Solin, A. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. In *ICLR*, 2017.

Schucany, W., Gray, H., and Owen, D. On bias reduction in estimation. *Journal of the American Statistical Association*, 66(335):524–533, 1971.

Seeger, M. Bayesian model selection for support vector machines, gaussian processes and other kernel classifiers. In *Advances in neural information processing systems*, pp. 603–609, 2000.

Shang, X., Zhu, Z., Leimkuhler, B., and Storkey, A. J. Covariance-controlled adaptive Langevin thermostat for large-scale bayesian sampling. In *Advances in Neural Information Processing Systems*, pp. 37–45, 2015.

Shekhovtsov, A. and Flach, B. Stochastic normalizations as bayesian learning. *arXiv preprint arXiv:1811.00639*, 2018.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4779–4783. IEEE, 2018.

Shen, K., Ju, Z., Tan, X., Liu, Y., Leng, Y., He, L., Qin, T., Zhao, S., and Bian, J. Naturalspeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers. *arXiv preprint arXiv:2304.09116*, 2023.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Simsekli, U., Sagun, L., and Gurbuzbalaban, M. A tail-index analysis of stochastic gradient noise in deep neural networks. *arXiv preprint arXiv:1901.06053*, 2019.

Singh, S. and Krishnan, S. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. *arXiv preprint arXiv:1911.09737*, 2019.

Skerry-Ryan, R., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., Weiss, R., Clark, R., and Saurous, R. A. Towards end-to-end prosody transfer for expressive speech synthesis with tacotron. In *international conference on machine learning*, pp. 4693–4702. PMLR, 2018a.

Skerry-Ryan, R., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., and Weiss, Ron J. Rob Clark, R. A. S. Towards End-to-End Prosody Transfer for Expressive Speech Synthesis with Tacotron. In *International Conference on Machine Learning*. PMLR, 2018b.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. How to train deep variational autoencoders and probabilistic ladder networks. In *33rd International Conference on Machine Learning (ICML 2016)*, 2016.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Sugita, Y. and Okamoto, Y. Replica-exchange molecular dynamics method for protein folding. *Chemical physics letters*, 314(1-2):141–151, 1999.

Sun, G., Zhang, Y., Weiss, R. J., Cao, Y., Zen, H., and Wu, Y. Fully-hierarchical fine-grained prosody modeling for interpretable speech synthesis. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6264–6268, 2020. doi: 10.1109/ICASSP40776.2020.9053520.

Sun, S., Chen, C., and Carin, L. Learning structured weight uncertainty in bayesian neural networks. In *Artificial Intelligence and Statistics*, pp. 1283–1292, 2017.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational Bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Swendsen, R. H. and Wang, J.-S. Replica Monte Carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

Tan, L. S. and Nott, D. J. Gaussian variational approximation with sparse precision matrices. *Statistics and Computing*, 28(2):259–275, 2018.

Tan, X. *Neural Text-to-Speech Synthesis*. Springer Nature, 2023.

Taylor, P. *Text-to-speech synthesis*. Cambridge university press, 2009.

Tieleman, T. and Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning, 2012.

Titsias, M. and Lázaro-Gredilla, M. Doubly stochastic variational bayes for non-conjugate inference. In *International conference on machine learning*, pp. 1971–1979, 2014.

Torresquintero, A., Teh, T. H., Wallis, C. G., Staib, M., Mohan, D. S. R., Hu, V., Foglianti, L., Gao, J., and King, S. ADEPT: A Dataset for Evaluating Prosody Transfer. In *Proc. Interspeech*, pp. 3880–3884, 2021. doi: 10.21437/Interspeech.2021-1610.

Tran, D., Dusenberry, M. W., Hafner, D., and van der Wilk, M. Bayesian Layers: A module for neural network uncertainty. In *Neural Information Processing Systems*, 2019.

Turner, R. and Sahani, M. *Two problems with variational expectation maximisation for time-series models*, pp. 109–130. Cambridge University Press, 2011.

Van Coile, B., Van Tichelen, L., Vorstermans, A., Jang, J., and Staessen, M. Protran: a prosody transplantation tool for text-to-speech applications. In *International Conference on Spoken Language Processing*. IEEE, 1994. URL `http://www.isca-speech.org/archive/icslp_1994/i94_0423.html`.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Virkar, Y., Federico, M., Enyedi, R., and Barra-Chicote, R. Improvements to prosodic alignment for automatic dubbing. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7543–7574, 2021. doi: 10.1109/ICASSP39728.2021.9414966.

Wang, C., Riviere, M., Lee, A., Wu, A., Talnikar, C., Haziza, D., Williamson, M., Pino, J., and Dupoux, E. VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 993–1003, Online, August 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.acl-long.80`.

Wang, C., Chen, S., Wu, Y., Zhang, Z., Zhou, L., Liu, S., Chen, Z., Liu, Y., Wang, H., Li, J., et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

Wang, Y., Stanton, D., Zhang, Y., Ryan, R.-S., Battenberg, E., Shor, J., Xiao, Y., Jia, Y., Ren, F., and Saurous, R. A. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *International Conference on Machine Learning*, pp. 5180–5189. PMLR, 2018.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

Wen, Y., Tran, D., and Ba, J. BatchEnsemble: Efficient ensemble of deep neural networks via rank-1 perturbation. 2019. Bayesian deep learning workshop 2019.

Weng, L. From autoencoder to beta-vae. *lilianweng.github.io*, 2018. URL `https://lilianweng.github.io/posts/2018-08-12-vae/`.

Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

Wilson, A. G. The case for Bayesian deep learning. *NYU Courant Technical Report*, 2019. Accessible at `https://cims.nyu.edu/~andrewgw/caseforbdl.pdf`.

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

Wolpert, D. H. and Wolf, D. R. Estimating functions of probability distributions from a finite set of samples. *Physical Review E*, 52(6):6841, 1995.

Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernández-Lobato, J. M., and Gaunt, A. L. Deterministic variational inference for robust bayesian neural networks. In *International Conference on Learning Representations (ICLR 2019)*, 2019.

Wu, Y., Tan, X., Li, B., He, L., Zhao, S., Song, R., Qin, T., and Liu, T.-Y. Adaspeech 4: Adaptive text to speech in zero-shot scenarios. *arXiv preprint arXiv:2204.00436*, 2022.

Yaida, S. Fluctuation-dissipation relations for stochastic gradient descent. *arXiv preprint arXiv:1810.00004*, 2018.

Yang, G. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

Zen, H., Tokuda, K., and Black, A. W. Statistical parametric speech synthesis. *speech communication*, 51(11):1039–1064, 2009.

Zhang, C., Butepage, J., Kjellstrom, H., and Mandt, S. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 2018a.

Zhang, C., Ren, Y., Tan, X., Liu, J., Zhang, K., Qin, T., Zhao, S., and Liu, T.-Y. DenoiSpeech: Denoising text to speech with frame-level noise modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7063–7067. IEEE, 2021.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*, 2017.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. *International Conference on Machine Learning*, 2018b.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient MCMC for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019a.

Zhang, Y., Cong, J., Xue, H., Xie, L., Zhu, P., and Bi, M. VISinger: Variational inference with adversarial learning for end-to-end singing voice synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7237–7241. IEEE, 2022.

Zhang, Y.-J., Pan, S., He, L., and Ling, Z.-H. Learning latent representations for style control and transfer in end-to-end speech synthesis. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6945–6949, 2019b. doi: 10.1109/ICASSP.2019.8683623.

Zhang, Y.-J., Pan, S., He, L., and Ling, Z.-H. Learning latent representations for style control and transfer in end-to-end speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6945–6949. IEEE, 2019c.

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, 2019.