Titre: Title:	Approximate Inference in Bayesian Neural Networks
Auteur: Author:	Nadhir Hassen
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Hassen, N. (2021). Approximate Inference in Bayesian Neural Networks [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/9899/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Approximate Inference in Bayesian Neural Networks

NADHIR HASSEN
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Décembre 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Approximate Inference in Bayesian Neural Networks

présenté par **Nadhir HASSEN**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Richard LABIB, président
Mario LEFEBVRE, membre et directeur de recherche
Irina RISH, membre et codirectrice de recherche
Pierre-Luc BACON, membre

DEDICATION

I dedicate my dissertation work to my family and many friends. I am especially grateful to my parents, whose loving words of encouragement and support for my tenacity ring in my ears. My two brothers have also been constant supporters and are very dear to me. I also dedicate this dissertation to the many colleagues who have become my friends and have supported me throughout my work at MILA. I will always appreciate the support they have provided, especially Xavier for helping me develop my technology skills, Kartik for many hours of proofreading, and Jean-Christophe for helping me to master leader dots!

RÉSUMÉ

Les réseaux de neurones profonds sont capables de résoudre de nombreux problèmes d'apprentissage automatique et aptes à atteindre des performances de pointe en raison de leur flexibilité, mais les modèles d'apprentissage profond sont difficiles à interpréter et souffrent d'un sur-ajustement statistique qui affecte leurs capacités de généralisation; en effet, ces derniers ont tendance à surestimer l'intervalle de confiance par rapport à leurs prédictions. Ceci peut être problématique pour les champs d'applications réels tels que les diagnostics médicaux ou les voitures autonomes. L'inférence bayésienne fournit des outils utiles pour s'attaquer à ces problèmes, mais cela a un coût, l'inférence bayésienne (exacte) pour les réseaux de neurones est dans la plupart des cas complexe dû à l'absence de forme analytique. L'apprentissage bayésien reste un bon choix pour concevoir des méthodes efficaces en fournissant une solution approximative. En effet, cette méthode combine inférence approximative et un cadre d'optimisation plus flexible. Cependant, l'efficacité des réseaux de neurones bayésiens est limitée à des distributions spécifiques et dans la majorité des cas la distribution a posteriori n'a pas de forme explicite.

Dans ce mémoire, nous abordons ces problèmes en démystifiant la relation entre l'inférence approximative et les méthodes d'optimisation à l'aide de la méthode de Gauss-Newton généralisée. Les réseaux de neuronnes bayésiens affichent de bons résultats en combinant la méthode de Gauss-Newton généralisée avec l'approximation de Laplace et gaussienne. Les deux méthodes calculent une approximation gaussienne de la distribution a posteriori, mais on ne sait pas comment elles affectent le modèle probabiliste sous-jacent. Les deux méthodes se basent sur un traitement rigoureux du modèle probabiliste sous-jacent mais l'interprétation de leurs résultats est moins claire. Nous cherchons à être en mesure de distinguer lorsqu'un modèle particulier échoue et la capacité de quantifier son incertitude. Nous avons constaté que la méthode de Gauss-Newton généralisée simplifie le modèle probabiliste sous-jacent et fournit un degré d'incertitude. En particulier, l'approximation de Laplace et l'approximation gaussienne fournissent une distribution a posteriori plus flexible qui peut être appliquée lorsque l'échantillon de données est assez grand. Dans ce travail, nous présentons une méthode d'inférence qui relie les deux approches. En fait, l'approximation gaussienne est considérée comme un concurrent direct de l'approximation de Laplace, fournit une inférence dans l'espace de fonction tandis que Laplace manifeste une inférence dans l'espace de paramètres. La combinaison de l'une ou l'autre à la méthode de Gauss-Newton doit être considérée comme une linéarisation locale du réseau de neurones bayésien. Ainsi, on obtient un modèle linéaire généralisé (GLM). Ce cadre permet de résoudre les problèmes courants de sous-ajustement

de l'approximation de Laplace. Plus intéressant même, nous sommes capables de faire la conversion d'un modèle GLM à un processus gaussien; ceci permet de faire le lien entre l'inférence dans l'espace de paramètres et l'inférence dans l'espace de fonctions dans le cadre de réseaux de neurones bayésiens.

ABSTRACT

Deep neural networks provide ways to tackle many real-world machine learning problems, achieving state-of-the-art performance due to their flexibility. However, deep learning models can be hard to interpret, sometimes suffer from overfitting, which affects their generalization capabilities, and tend to overstate the confidence of their predictions. This can be problematic for real-world applications such as medical diagnostics or self-driving cars. Bayesian pragmatism provides useful tools to tackle these issues, but it comes at a cost: the exact Bayesian inference appropriate to a neural network is often intractable. Bayesian deep learning remains a good choice to design efficient methods by providing an approximate solution; combining as it does approximate inference and a scalable optimization framework. However, the practical effectiveness of Bayesian neural networks is limited by the need to specify meaningful prior distributions, and by the intractability of posterior inference.

In this thesis, we address these issues by attempting to demystify the relationship between approximate inference and optimization approaches through the generalized Gauss–Newton method. Bayesian deep learning yields good results, combining Gauss–Newton with Laplace and Gaussian variational approximation. Both methods compute a Gaussian approximation to the posterior; however, it remains unclear how these methods affect the underlying probabilistic model and the posterior approximation. Both methods allow a rigorous analysis of how a particular model fails and the ability to quantify its uncertainty. We found that the generalized Gauss–Newton method simplifies the underlying probabilistic model and provides a meaningful uncertainty quantification. In particular, the Laplace and Gaussian variational approximations provide a tractable and scalable approach to posterior approximation, applicable to large datasets. In this work, we use the Bayesian approach to infer neural networks based on two approximate inference techniques. In fact, the Gaussian variational approximation is considered as a direct competitor to Laplace approximation, providing function-space inference while Laplace performs weight-space inference. The combination of either method with Gauss–Newton should be understood as a local linearization of the Bayesian neural network, which becomes a generalized linear model (GLM). This approach enables us to resolve common underfitting problems with the Laplace approximation; the conversion to Gaussian processes enables inference schemes for Bayesian neural networks in function space.

Key words: *Bayesian optimization, deep learning, variational inference, Laplace approximation, neural networks.*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

Symbol	Description
$\mathbb{R}, \mathbb{R}_+$	the sets of real and positive real numbers
x	scalar variable
$\mathbf{L}$	matrix variable with scalar entries $\mathbf{L}_{ij}$
$\mathbf{f}(\mathbf{x}; \mathbf{w})$	function mapping $\mathbf{x}$ to some output parameterized by $\mathbf{w}$, for example a neural network
$\nabla_{\mathbf{w}} \mathbf{f}(\mathbf{x}; \mathbf{w})$	gradient, with individual entries $[\nabla_{\mathbf{w}} \mathbf{f}(\mathbf{x}; \mathbf{w})]_i := \frac{\partial \mathbf{f}(\mathbf{x}; \mathbf{w})}{\partial \mathbf{w}_i}$
$\nabla^2_{\mathbf{ww}} \mathbf{f}(\mathbf{x}; \mathbf{w})$	Hessian, with entries $[\nabla_{\mathbf{w}} \mathbf{f}(\mathbf{x}; \mathbf{w})]_{ij} := \frac{\partial^2 \mathbf{f}(\mathbf{x}; \mathbf{w})}{\partial \mathbf{w}_i \partial \mathbf{w}_j}$
$\nabla_{\mathbf{w}} \mathbf{f}(\mathbf{x}; \mathbf{w}_\star)$	gradient evaluated at $\mathbf{w} = \mathbf{w}_\star$
$\nabla^2_{\mathbf{ww}} \mathbf{f}(\mathbf{x}; \mathbf{w}_\star)$	Hessian evaluated at $\mathbf{w}_\star$
$\mathbf{f}^{\mathbf{w}_\star}_{\text{lin}}(\mathbf{x}; \mathbf{w})$	linearized neural network evaluated at $\mathbf{w}_\star$
$\langle ., . \rangle$	scalar, vector, or Frobenius inner product
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathbf{w}$, distributed according to $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Acronym	Description
GGN	generalized Gauss–Newton
GVA	Gaussian variational approximation
VI	variational inference
GP	Gaussian process
GLM	generalized linear model
BNN	Bayesian neural network (naive BNN)
BLR	Bayesian linear regression (model)
VOGGN	variational online generalized Gauss–Newton (algorithm)

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Bayesian methods are very powerful tools for inferring hidden representations from data. A significant body of work has extended their application to deep learning, but they remain impractical and rarely match the performances of standard methods. Two major challenges have been related to the success of application of deep learning; scalability which is the ability to make generalization on large datasets and tractability which measures the efficiency of the model. Bayesian neural networks (BNNs) have the potential to combine the scalability and predictive performance of neural networks with the principle of Bayesian uncertainty modelling [1; 2]. However, the practical effectiveness of BNNs is limited by our ability to specify meaningful prior distributions and by the intractability of its posterior inference. Bayesian neural networks are probabilistic models in which the likelihood is parameterized by a neural network and the prior is a distribution over the neural network parameters. Bayesian inference in these models is particularly challenging and combining Bayesian estimation and posterior optimization has therefore played a major role in tackling these problems [3]. Approximate Bayesian inference for probabilistic neural network models remains a good alternative. Specifically, instead of obtaining an exact posterior of the Bayesian inference problem, an approximation to the posterior is constructed. The use of approximate inference techniques with BNNs affords parameter uncertainties that allow predictive uncertainties while maintaining the computational and performance advantages of deep learning. A key concept of Bayesian deep learning algorithms is to combine scalable optimization and probabilistic inference. The combination of the generalized Gauss–Newton method and Gaussian posterior approximations make Bayesian deep learning competitive with traditional deep learning techniques, but their connection is still unclear. In this work we address this issue, showing how Gaussian posterior approximations and the generalized Gauss–Newton method allow to better understand these algorithms and improve them to a larger set of problems such as out-of-distribution detection and uncertainty quantification.

1.1 Approximate Inference in Neural Networks

In Bayesian learning we deal with the posterior distribution which captures global information about the data; by contrast, point estimates such as the maximum a posteriori (MAP) estimate carry only local information. In particular, in the case of the regression task, Bayesian inference is preferred and moreover provides additional advantages over MAP estimates. First, it enables the uncertainty of predictions to be quantified. Second, the Bayesian

formalism has the capability to capture the marginal likelihood of the data, which allows us to compare different models and select the one that best explains the data. Interestingly, we can combine Bayesian formalism with MAP estimation. Formally, in deep learning we are given a neural network denoted by $\mathbf{f}(\mathbf{x};\mathbf{w}) : \mathbb{R}^D \times \mathbb{R}^P \to \mathbb{R}^K$ that maps inputs $\mathbf{x} \in \mathbb{R}^D$ to an output $\mathbf{f} \in \mathbb{R}^K$ contingent on parameters $\mathbf{w} \in \mathbb{R}^P$. Let's consider the case of a supervised learning task with a pair-dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ comprising a set of inputs $\mathbf{x}_n \in \mathbb{R}^D$ and labels $\mathbf{y}_n \in \mathbb{R}^K$. We assume the data points are identically distributed and independent. In Bayesian inference we use Bayes' rule to compute the posterior distribution $p(\mathbf{w}|\mathcal{D})$ over the parameters as follows:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{\int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}} = \frac{p(\mathcal{D},\mathbf{w})}{p(\mathcal{D})} \propto p(\mathcal{D},\mathbf{w});$$

here $p(\mathcal{D})$ denotes the marginal likelihood, which plays the role of normalization constant and in most cases is intractable. This expression makes apparent the combination of prior belief $p(\mathbf{w})$ and the likelihood of observing the data label $\mathbf{y}$ denoted by $p(\mathbf{y}|\mathbf{x};\mathbf{w})$. For a neural network model, the likelihood is given by

$$p(\mathcal{D}|\mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n;\mathbf{w})).$$

The goal is to optimize the objective function by the MAP estimate; in practice we maximize the following log joint distribution known as the *empirical risk minimization*. From Bayes' rule we have

$$\begin{aligned}\mathbf{w}_{\text{MAP}} &= \arg\max_{\mathbf{w}} \log p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n;\mathbf{w})) + \log p(\mathbf{w});\end{aligned}$$

this objective exhibits a decomposition of the likelihood term that depends on data points and the prior, the latter acting as a regularizer. To enable a tractable form, the likelihood can be restricted to the exponential family of distributions due to their desirable properties for deep learning inference.

1.2 Approximate Inference in Function Space

Alternatively, we can apply Bayesian inference in the function space instead of parameter space by making use of Gaussian processes. Gaussian processes (GPs) are stochastic processes that provide a flexible non-parametric framework to perform probabilistic inference [4]. A

Gaussian process models a distribution over function values such that any combination of a finite number of dimensions is Gaussian distributed. Remarkable properties of GPs enable tractable posterior inference (closed-form for Gaussian likelihoods) and calibrated uncertainty estimates. For a simple regression problem, the inputs $\mathbf{x}$ and the output y are generated by an unknown mapping $\mathbf{f}$ with Gaussian-distributed noise $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. We consider the standard linear regression model with Gaussian noise $y = f(\mathbf{x}) + \varepsilon$. A remarkable property of GPs is that, for any finite number of inputs $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T$, the marginal distribution of the function values $\mathbf{f} = [\mathbf{f}(\mathbf{x}_1), \ldots, \mathbf{f}(\mathbf{x}_n)]^T$ for $\mathbf{f} \in \mathbb{R}^K$, is a multivariate Gaussian distribution. The goal is to compute the posterior distribution over the function $\mathbf{f}(\mathbf{x})$ evaluated at arbitrary test inputs $\mathbf{x}$. First we need to assume that $\mathbf{f} \sim \mathcal{GP}(\mathbf{m}(\mathbf{x}), \boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\lambda}))$ is a Gaussian process characterised by its mean $\mathbf{m}(\mathbf{x})$ and covariance function $\boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\lambda})$. We apply Bayesian formalism by modelling the joint probability of the system $\{\mathbf{y}, \mathbf{X}, \mathbf{f}\}$ as follows

$$\begin{aligned} \mathbf{f}|\mathbf{X}, \boldsymbol{\lambda} &\sim \mathcal{GP}(\mathbf{m}(\mathbf{x}), \mathbf{K}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\lambda})), \\ p(\mathbf{y}|\mathbf{f}, \mathbf{X}; \boldsymbol{\lambda}) &= p(\mathbf{f}|\mathbf{X}; \boldsymbol{\lambda}) \prod_{n=1}^{N} p(y_n|\mathbf{f}, \mathbf{x}_n), \end{aligned} \tag{1.2.1}$$

where $\mathbf{K}(.,.;\boldsymbol{\lambda})$ is the covariance matrix induced by the covariance function $\boldsymbol{\kappa}(.,.;\boldsymbol{\lambda})$ evaluated at every pair of inputs, and the vector parameter $\boldsymbol{\lambda} = \{\lambda_j\}_{j=1}^{J}$ stores the parameters of the corresponding covariance functions. The mean prior $\mathbf{m}(\mathbf{x})$ is generally set to zero and the covariance function $\mathbf{K} \in \mathbb{R}^{N \times N}$ is a positive definite kernel which depends on the hyperparameters $\boldsymbol{\lambda}$. The likelihood function denoted by $p(\mathbf{y}|\mathbf{f}, \mathbf{x})$ can take different forms depending on whether we consider a regression or a classification problem. Given such a prior distribution, the goal of Bayesian inference is to compute the posterior distribution over the function $\mathbf{f}(\mathbf{x}_n)$ evaluated at arbitrary test inputs $\mathbf{x}_n$. When the likelihood is Gaussian distributed, the posterior distribution takes a convenient closed-form solution on account of its conjugacy prior; therefore the predictive distribution at a test location $\mathbf{x}_\star$ can be written in closed form as follows:

$$\begin{aligned} \mathbf{f}(\mathbf{x}_\star)|\mathbf{y}, \mathbf{X} \sim \mathcal{N}\Bigg(& \mathbf{K}(\mathbf{x}_\star, \mathbf{X}; \boldsymbol{\lambda}) \left(\mathbf{K}(\mathbf{X}, \mathbf{X}; \boldsymbol{\lambda}) + \sigma_n^2 \mathbf{I}_n\right)^{-1} (\mathbf{y} - \mathbf{m}(\mathbf{X})), \\ & \boldsymbol{\kappa}(\mathbf{x}_\star, \mathbf{x}_\star; \boldsymbol{\lambda}) - \mathbf{K}(\mathbf{x}_\star, \mathbf{X}; \boldsymbol{\lambda}) \left(\mathbf{K}\left(\mathbf{X}, \mathbf{X}; \boldsymbol{\lambda}\right) + \sigma_n^2 \mathbf{I}_n\right)^{-1} \mathbf{K}\left(\mathbf{X}, \mathbf{x}_\star; \boldsymbol{\lambda}\right)\Bigg). \end{aligned}$$

To determine the hyperparameters $\boldsymbol{\lambda}$ and σ_n^2, we require the marginal likelihood of the data

$$p(\mathcal{D}|\mathbf{X}, \boldsymbol{\lambda}, \sigma_n^2) = \int p(\mathbf{y}|\mathbf{f}, \sigma_n^2) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\lambda}) d\mathbf{f}$$

In the case of multi-class classification, we construct k independent Gaussian process functions which are passed through a *softmax* function. Although we no longer have a closed-form marginal likelihood function, it can generally be approximated by a Monte Carlo approximation. Another limitation of GPs is that even though an exact posterior is available, scaling to large datasets is computationally expensive because of the complexity cost of the marginal likelihood computation, which is due to the cost of matrix inversion scaling as $\mathcal{O}(N^3)$. To be able to scale to large datasets we resort to sparse approximation, which reduces the computational cost but could result in lesser-quality estimates.

Sparse Approximation of GPs

To reduce the computation cost, sparse approximation methods can be applied; these consist of choosing a small number M of "inducing" points, where $M \ll N$. A variety of sparse approximations have been proposed in [5] and they mostly differ in how the selection is made to process the matrix inversion. We describe below a method presented by [6] that considers a subset of M function values $\mathbf{u} = \{\mathbf{u}_n\}_{n=1}^M$ from the infinite-dimensional object $\mathbf{f} = \{\mathbf{f}_{\neq \mathbf{u}}, \mathbf{u}\}$ induced by a set of inducing points $\mathbf{Z} = \{\mathbf{z}_n\}_{n=1}^M$. We can now rewrite equation (1.2.1) as follows:

$$p(\mathbf{y}, \mathbf{f}|\mathbf{X}; \boldsymbol{\lambda}) = p(\mathbf{u}|\mathbf{Z}; \boldsymbol{\lambda}) p(\mathbf{f}_{\neq \mathbf{u}}|\mathbf{X}, \mathbf{u}, \mathbf{Z}; \boldsymbol{\lambda}) p(\mathbf{y}|\mathbf{f}, \mathbf{X}).$$

The key idea is to approximate the posterior $\mathbf{f}$ by restricting the variational approximation. That is, we construct a factored variational distribution such that $q(\mathbf{f}) = q(\mathbf{u}) p(\mathbf{f}_{\neq \mathbf{u}}|\mathbf{X}, \mathbf{u}, \mathbf{Z}; \boldsymbol{\lambda})$ in order to obtain a tractable expression of the variational lower bound to the log marginal likelihood, given by

$$\begin{aligned} \mathcal{L}(q) :&= \mathbb{E}_{q(\mathbf{f})} \left[\log \frac{p(\mathbf{u}|\boldsymbol{\lambda}) p(\mathbf{u}|\mathbf{Z}; \boldsymbol{\lambda}) p(\mathbf{y}|\mathbf{f}, \mathbf{X})}{q(\mathbf{u}) p(\mathbf{u}|\mathbf{Z}; \boldsymbol{\lambda})} \right] \\ &= \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{f})} \left[\log p(\mathbf{y}_n|\mathbf{f}, \mathbf{x}_n) \right] - D_{KL} \left(q(\mathbf{u}) || p(\mathbf{u}|\boldsymbol{\lambda}) \right), \end{aligned} \tag{1.2.2}$$

where D_{KL} denotes the *Kullback–Leibler* (KL) divergence measure. This formulation enables us to apply stochastic variational inference as proposed in [7], in which the authors showed that by sub-sampling the likelihood term they were able to obtain noisy but unbiased estimates of the hyperparameters. Interestingly, Gaussian processes can provide function space inference entailing rich structures, which allows us to illustrate the connection from weight space to function space and thus enrich Bayesian learning.

1.3 Thesis Outline

In this thesis we establish a connection between approximate inference and optimization methods that accomplishes a twofold objective: scalability and tractability. More specifically, we apply Laplace and the generalized Gauss–Newton (GGN) approximation, which give rise to the generalized linear model (GLM) due to the linearizing property of the GGN. Subsequently, we will be able to specify a weight space Bayesian inference and equivalently identify a Gaussian process model for the neural network in function space. This formulation yields a better understanding of uncertainty and can potentially enhance our understanding of our probabilistic model.

In Chapter 2 we give the necessary background on Laplace approximation and the generalized Gauss–Newton method for probabilistic neural network models. We show how the combination of the Laplace approximation with GGN can efficiently optimize the neural network. Then, we introduce Gaussian variational approximation and show how this method differs from Laplace approximation, albeit both lead to a GLM. Chapter 3 makes the connection between approximate inference for GLM neural networks and Gaussian process models. We discuss how the conversion from weight space to function space allows scalability and tractability of neural networks. Further, we show how to compute the posterior predictive and marginal likelihood approximations of neural network models. Chapter 4 sets out the impact of the generalized Gauss–Newton method on variational inference; we also present the computational considerations involved in scaling the Bayesian inference to large datasets. Chapter 5 reports experiments that explain and complement the theoretical connections: we show how to tune hyperparameters using the marginal likelihood and the posterior predictive. We also extend our analysis to different covariance matrix structures. Further, we illustrate how these methods can solve common deep learning problems such as uncertainty quantification and out-of-distribution detection. Finally, we highlight interesting future directions in Chapter 6 and conclude this work in Chapter 7.

CHAPTER 2 BAYESIAN NEURAL NETWORKS

2.1 Introducing Bayesian Neural Networks

A Bayesian neural network (BNN) can be defined as a stochastic artificial neural network that is trained using Bayesian inference without restricting to any deep neural network architecture. The first step is to choose a likelihood and a prior distribution. Due to their desirable theoretical properties, generalized linear models (GLMs) are a family of likelihoods often used in deep learning that allow tractable inference and provide efficient gradient computations [8]. Let's consider a response random vector $\mathbf{Y}$; we define an invertible link function $\mathbf{g}$ that expresses the mean of the response vector as $\mathbb{E}[\mathbf{Y}] = \mathbf{g}^{-1}(\mathbf{f})$. A generic exponential family can have the *natural form*

$$p(\mathbf{y}|\mathbf{f}) = h(\mathbf{y})e^{\langle T(\mathbf{y}),\mathbf{f}\rangle - A(\mathbf{f})}, \tag{2.1.1}$$

where $T(\mathbf{y})$ is a sufficient statistic, $h(\mathbf{y})$ is a base measure, $A(\mathbf{f})$ is the log-partition and $\mathbf{f}$ is the vector of natural parameters. It is common to set $T(\mathbf{y}) = \mathbf{y}$ as the identity, so that equation (2.1.1) yields the form $h(\mathbf{y})e^{\langle \mathbf{y},\mathbf{f}\rangle - A(\mathbf{f})}$. This formulation makes the derivatives of the log-likelihood with respect to the function $\mathbf{f}$ convenient. Therefore we can directly relate $\mathbf{f}$ to the moments of $\mathbf{Y}$. The first derivative can be written as a residual between the observed label and the mean of the response variable, identified as the link function $\mathbf{g}$. The second derivative is directly related to the variance of the response variable. For an exponential family distribution $p(\mathbf{y}|\mathbf{f})$ of the form in (2.1.1), [4] generalizes this result by providing the first and the second derivative of the log likelihood as follows:

$$\begin{aligned} \boldsymbol{R}(\mathbf{f}) &:= \nabla_{\mathbf{f}} \log p(\mathbf{y}|\mathbf{f}) = \mathbf{y} - \nabla_{\mathbf{f}} A(\mathbf{f}) = \mathbf{y} - \mathbb{E}[\mathbf{Y}] = \mathbf{y} - \mathbf{g}^{-1}(\mathbf{f}) \\ -\boldsymbol{\Lambda}(\mathbf{f}) &:= \nabla^2_{\mathbf{ff}} \log p(\mathbf{y}|\mathbf{f}) = -\nabla^2_{\mathbf{ff}} A(\mathbf{f}) = -\mathrm{Var}(\mathbf{Y}), \end{aligned}$$

where the inverse-link function $\mathbf{g}^{-1}(\mathbf{f})$ gives the mean of $\mathbf{Y}$. The residual is denoted by $\boldsymbol{R}(\mathbf{f}) \in \mathbb{R}^K$ and $\boldsymbol{\Lambda}(\mathbf{f}) \in \mathbb{R}^{K\times K}$ denotes the Hessian of the negative log likelihood, also termed the noise precision. For the choice of the prior, multiple techniques have been proposed depending on the nature of the task. Since we are dealing with deep neural networks, we restrict ourselves to a multivariate Gaussian which allows a connection to be made with Laplace and Gaussian variational approximation [9], thereby illustrating the scalablility and tractablility of the predictive distribution. In deep learning, it is common to use Gaussian and Bernoulli likelihoods; when optimizing the objective function, the Gaussian likelihood yields

a least-square loss in the regression case [4]. For the classification case, we use Bernoulli and categorical distributions, for which the *cross-entropy* is used as a loss function. A common choice of the prior is a multivariate Gaussian, because in the context of the MAP estimation a Gaussian prior corresponds to an ℓ_2 regularization (weight decay). Therefore, we assume the prior of $\mathbf{w}$ has the following distribution:

$$\mathbf{w} \sim p(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{S}_0),$$

where the mean is denoted by $\boldsymbol{\mu}_0 \in \mathbb{R}^P$ and the covariance by $\mathbf{S}_0 \in \mathbb{R}^{P \times P}$. A common choice is to set the mean to zero and the covariance to a diagonal matrix. For a probabilistic neural network model, the exact Bayesian inference is in most cases intractable. One way to deal with this issue is to resort to approximate inference methods that enable computation of the marginal likelihood. Approximate inference produces a scalable and tractable form based on Laplace approximation or the Gaussian variational approximation. Both techniques are constructed on a Gaussian approximation to the true posterior distribution. The Gaussian variational approximation consists of maximizing the evidence lower bound of the log marginal likelihood; this method is popular because it better captures the shape of the posterior [10]. In contrast, the Laplace approach approximates the marginal likelihood by using a second-order approximation of the joint distribution (or log joint distribution). We consider the posterior approximation $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \approx p(\mathbf{w}|\mathcal{D})$, where $\boldsymbol{\mu} \in \mathbb{R}^P$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{P \times P}$ are Gaussian parameters. In the next section we present both methods.

2.2 The Laplace Approximation

The Laplace approximation fits a Gaussian distribution locally at the MAP estimate. This approximation is divided in two steps: first, $\mathbf{w}_{\text{MAP}}$ is obtained through optimization, and then the log joint distribution is approximated using the second-order Taylor expansion. One advantage to this method, which provides an approximation to both the posterior and the marginal likelihood, is that the marginal likelihood is obtained in closed form. Since $\mathbf{w}_{\text{MAP}}$ is a minimizer, the gradient of the log joint with respect to $\mathbf{w}$ is equal to zero, and the second-order Taylor approximation evaluated at $\mathbf{w}_{\text{MAP}}$ is

$$\log p(\mathcal{D}, \mathbf{w}) \approx \log p(\mathcal{D}, \mathbf{w}_{\text{MAP}}) + \frac{1}{2} \left(\mathbf{w} - \mathbf{w}_{\text{MAP}}\right)^T \nabla^2_{\mathbf{ww}} \log p(\mathcal{D}, \mathbf{w}_{\text{MAP}}) \left(\mathbf{w} - \mathbf{w}_{\text{MAP}}\right);$$

following [10], we can therefore obtain the Laplace approximation to the marginal likelihood in a closed form as follows:

$$p(\mathcal{D}) \approx p(\mathcal{D}, \mathbf{w}_{\text{MAP}})(2\pi)^{P/2}|\nabla^2_{\mathbf{ww}} \log p(\mathcal{D}, \mathbf{w}_{\text{MAP}})|^{-1/2},$$

where $|.|$ denotes the determinant. We are now able to obtain the distribution

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}, \mathbf{w})}{p(\mathcal{D})} \approx p(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where we have identified a Gaussian form of the Laplace approximation with

$$\boldsymbol{\mu} = \mathbf{w}_{\text{MAP}} \quad \text{and} \quad \boldsymbol{\Sigma}^{-1} = -\nabla^2_{\mathbf{ww}} \log p(\mathcal{D}, \mathbf{w}_{\text{MAP}}).$$

The Laplace approximation is well suited to deep learning optimizers, because we only need to find a MAP estimate and compute the Hessian of the log joint distribution at that estimate. However, this method is not scalable in the case of large networks due to the computational cost of the Hessian, and in fact there is no guarantee that the Hessian at the MAP is necessarily positive definite. To tackle this problem, the Gauss–Newton method ensures an invertible Hessian approximation [11; 12]. In the next section we discuss the Gaussian variational approximation and introduce the generalized Gauss–Newton method.

Variational Approximation as a Competitor to Laplace Approximation

When the function value takes the form of a neural network, the variational approximation problem is restricted to optimizing the KL-divergence of the true posterior $p(\mathbf{w}|\mathcal{D}) :\approx q(\mathbf{w})$. In particular, when the approximate distribution belongs to the mean-field family $\mathcal{Q}$, then the variational approximation consists of minimizing the KL-divergence. Since the KL-divergence is non-negative, we have

$$\begin{aligned} q(\mathbf{w}) &= \arg\min_{q \in \mathcal{Q}} D_{KL}\left[q(\mathbf{w})||p(\mathbf{w}|\mathcal{D})\right] \\ &= \arg\min_{q \in \mathcal{Q}} \int q(\mathbf{w}) \log\left(\frac{q(\mathbf{w})}{p(\mathcal{D}, \mathbf{w})}\right) d\mathbf{w} + \log p(\mathcal{D}) \\ &\geq \arg\min_{q \in \mathcal{Q}} \int q(\mathbf{w}) \log\left(\frac{q(\mathbf{w})}{p(\mathcal{D}, \mathbf{w})}\right) d\mathbf{w} + \mathbb{E}_q\left[\log \frac{p(\mathcal{D}, \mathbf{w})}{q(\mathbf{w})}\right]. \end{aligned}$$

Gaussian variational approximation restricts the mean-field family to multivariate Gaussian distributions (often termed variational inference, or VI) parametrized by its mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Therefore, the *evidence lower bound* (ELBO) for a probabilistic neural network

model with Gaussian prior can be decomposed into an expected log likelihood denoted by $\mathcal{L}_{\text{lik}}(q(\mathbf{w}))$ and a KL-divergence term denoted by $\mathcal{L}_{\text{KL}}(q(\mathbf{w}))$. The lower bound is given by

$$\begin{aligned}\log p(\mathcal{D}) &\geq \mathcal{L}_{\text{KL}}(q(\mathbf{w})) + \mathcal{L}_{\text{lik}}(q(\mathbf{w})) \\ &= \mathbb{E}_q\left[\log p(\mathcal{D}|\mathbf{w})\right] - D_{KL}\left[q(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\Sigma})||p(\mathbf{w}|\mathcal{D})\right] := \mathcal{L}(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\Sigma}); \end{aligned} \tag{2.2.1}$$

since both prior and approximate distribution are Gaussian, we can obtain a closed form of the KL-divergence term.

2.3 Differences Between Laplace and Variational Gaussian Approximation

In this section we highlight the differences between Laplace approximation and the variational approximation when restricted to a Gaussian family. The stationary conditions for the variational Gaussian approximation can be obtained by optimizing the parameters of the ELBO; for a probabilistic neural network model with Gaussian prior, the gradients of $\mathcal{L}(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\Sigma})$ with respect to its (variational) parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by

$$\nabla_{\boldsymbol{\mu}}\mathcal{L}(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \nabla_{\boldsymbol{\mu}}\Big(\mathbb{E}_q\left[\log p(\mathcal{D}|\mathbf{w})\right] - D_{KL}\left[q(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\Sigma})||p(\mathbf{w}|\mathcal{D})\right]\Big)$$

and

$$\nabla_{\boldsymbol{\Sigma}}\mathcal{L}(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \nabla_{\boldsymbol{\Sigma}}\Big(\mathbb{E}_q\left[\log p(\mathcal{D}|\mathbf{w})\right] - D_{KL}\left[q(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\Sigma})||p(\mathbf{w}|\mathcal{D})\right]\Big).$$

[12] and [13] respectively provided simplified derivatives for the expected log likelihood, establishing a close connection to second-order approximation which can be useful for deep learning optimizers. Therefore the expected log likelihood ELBO term can be simplified using the derivatives with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of the expected log likelihood. This formulation reduces determining the gradient of the variational parameters to a sampling process that takes the following form:

$$\nabla_{\boldsymbol{\mu}}\mathbb{E}_{q(\mathbf{w})}\left[\log p(\mathcal{D}|\mathbf{w})\right] = \mathbb{E}_{q(\mathbf{w})}\left[\nabla_{\mathbf{w}}\log p(\mathcal{D}|\mathbf{w})\right] \tag{2.3.1}$$

and

$$\nabla_{\boldsymbol{\Sigma}}\mathbb{E}_{q(\mathbf{w})}\left[\log p(\mathcal{D}|\mathbf{w})\right] = \frac{1}{2}\mathbb{E}_{q(\mathbf{w})}\left[\nabla^2_{\mathbf{ww}}\log p(\mathcal{D}|\mathbf{w})\right]. \tag{2.3.2}$$

[14] showed that the equalities in (2.3.1) and (2.3.2) are also applicable for the joint distribution; we can therefore determine our stationary conditions using these equalities. The condition for the mean is trivial, while the condition for the inverse covariance consists of differentiating with respect to the variational parameters such that $\nabla_{\boldsymbol{\Sigma}}\mathcal{L}(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\Sigma}) = 0$, which yields

$$\boldsymbol{\Sigma}^{-1} = 2\nabla_{\boldsymbol{\Sigma}}\mathbb{E}_{q(\mathbf{w})}\left[-\log p(\mathcal{D},\mathbf{w})\right]. \tag{2.3.3}$$

We obtain the stationarity conditions as follows:

$$\mathbb{E}_{q(\mathbf{w})}\left[\nabla_{\mathbf{w}}\log p(\mathcal{D},\mathbf{w})\right] = 0 \quad \text{and} \quad \boldsymbol{\Sigma}^{-1} = \mathbb{E}_{q(\mathbf{w})}\left[-\nabla^2_{\mathbf{ww}}\log p(\mathcal{D},\mathbf{w})\right]. \tag{2.3.4}$$

Similarly, following [14], the optimality conditions for the Laplace approximation are given by

$$\nabla_{\mathbf{w}}\log p(\mathcal{D},\boldsymbol{\mu}) = 0 \quad \text{and} \quad -\nabla^2_{\mathbf{ww}}\log p(\mathcal{D},\boldsymbol{\mu}) = \boldsymbol{\Sigma}^{-1}. \tag{2.3.5}$$

Despite the computational framework, the Gaussian variational approximation is considered a more powerful method than Laplace approximation since the inference is performed in function space. The main difference between the approximations in equations (2.3.5) and (2.3.4) is that Laplace approximation is only defined locally at the MAP estimate while variational approximation holds globally. Note that due to the stochasticity in the parameters, the variational approximation provides a more global view of the joint distribution; see [14] for a more detailed analysis.

2.4 Approximation with the Generalized Gauss-Newton Method

Training neural networks using approximate Bayesian inference is computational expensive, due to the computation of second-order derivatives of the log likelihood. Generalized Gauss–Newton (GGN) approximation is often used for the Laplace and Gaussian variational approximation since it provides a positive semi-definite approximation to the Hessian. More scalable GGN techniques have been proposed, including diagonal [9] or Kronecker factorization [15]. In the case of probabilistic neural network models, we have defined a generalized linear model (GLM) likelihood as $\log p(\mathcal{D}|\mathbf{w}) = \sum_{n=1}^{N}\log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n;\mathbf{w}))$. We obtain the MAP estimate by optimizing $\mathbf{w}_{\text{MAP}}$. In deep learning we generally have a loss function $\ell(\mathbf{y},\mathbf{f}(\mathbf{x}_n;\mathbf{w}))$; the Bayesian learning problem is a generalization of Bayesian inference which requires a probabilistic model [16], and when the loss corresponds to the log of a probability distribution

$\ell(\mathbf{y}, \mathbf{f}(\mathbf{x}_n; \mathbf{w})) = -\log p(\mathbf{y}|\mathbf{f}(\mathbf{x}_n; \mathbf{w}))$, then the objective function takes the form

$$\min_{q \in \mathcal{Q}} -\mathbb{E}_{q(\mathbf{w})} \left[\log p(\mathcal{D}|\mathbf{w})\right] + D_{KL} \left[q(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma})||p(\mathbf{w}|\mathcal{D})\right], \tag{2.4.1}$$

known as *empirical risk minimization*. The solution $q(\mathbf{w})$ allows us to identify the posterior distribution of a probabilistic model with likelihood $p(\mathbf{y}|\mathbf{f}(\mathbf{x}_n; \mathbf{w}))$ and prior $p(\mathbf{w})$ [16]. If the approximation family $\mathcal{Q}$ contains the true posterior distribution, then the variational approximation is exact and reduces the KL-divergence to zero. With this setting, the likelihood acts as a loss per data point and the prior can be understood as a regularizer. To optimize $\log p(\mathcal{D}|\mathbf{w})$ we take the first and second derivatives with respect to the parameter vector $\mathbf{w}$. Applying the chain rule we can first differentiate with respect to $\mathbf{f}$ and then with respect to the parameters, as follows:

$$[\boldsymbol{J}(\mathbf{x}; \mathbf{w})]_{ij} = \frac{\partial \mathbf{f}_i(\mathbf{x}; \mathbf{w})}{\partial \mathbf{w}_j} \quad \text{and} \quad [\boldsymbol{H}(\mathbf{x}; \mathbf{w})]_{ijk} = \frac{\partial^2 \mathbf{f}_i(\mathbf{x}; \mathbf{w})}{\partial \mathbf{w}_j \partial \mathbf{w}_k},$$

where $\boldsymbol{J}(\mathbf{x}; \mathbf{w}) \in \mathbb{R}^{K \times P}$ is the Jacobian of $\mathbf{f}(\mathbf{x}; \mathbf{w})$, which we assume to be twice differentiable with respect to the parameters. Similarly, the Hessian of the second derivatives corresponds to $\boldsymbol{H}(\mathbf{x}; \mathbf{w}) \in \mathbb{R}^{K \times P \times P}$. When $p(\mathbf{y}|\mathbf{f})$ is an exponential distribution of the form of $p(\mathbf{y}|\mathbf{f}) = h(\mathbf{y}) \exp(\langle \mathbf{y}, \mathbf{f} \rangle - A(\mathbf{f}))$ we can use the properties of the first and second derivatives of the GLM log likelihood, obtaining the following gradient expression:

$$\nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}; \mathbf{w})) = \boldsymbol{J}(\mathbf{x}; \mathbf{w})^T \nabla_{\mathbf{f}} \log p(\mathbf{y}|\mathbf{f}). \tag{2.4.2}$$

This expression is easy to compute even with complex neural network architecture. The Hessian of the log likelihood takes the form

$$\begin{aligned} \nabla_{\mathbf{ww}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}; \mathbf{w})) &= \boldsymbol{H}(\mathbf{x}; \mathbf{w})^T \nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f}) - \boldsymbol{J}(\mathbf{x}; \mathbf{w})^T \nabla^2_{\mathbf{ff}} \log p(\mathbf{y}|\mathbf{f}) \boldsymbol{J}(\mathbf{x}; \mathbf{w}) \\ &= \boldsymbol{H}(\mathbf{x}; \mathbf{w})^T \nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f}) - \boldsymbol{J}(\mathbf{x}; \mathbf{w})^T \boldsymbol{\Lambda}(\mathbf{f}) \boldsymbol{J}(\mathbf{x}; \mathbf{w}), \end{aligned} \tag{2.4.3}$$

where $\boldsymbol{\Lambda}(\mathbf{f}) := \nabla^2_{\mathbf{ff}} \log p(\mathbf{y}|\mathbf{f})$. Equation (2.4.3) is intractable, however, first because we need to differentiate the neural network twice, and second, in some neural network architectures the second derivative for particular activation functions is not defined everywhere [17]. To solve this problem, we can use the GGN approximation to the Hessian to remove the intractable term, so that the Hessian becomes

$$\begin{aligned} \nabla^2_{\mathbf{ww}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}; \mathbf{w}) &= \boldsymbol{H}(\mathbf{x}; \mathbf{w})^T \nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f}) - \boldsymbol{J}(\mathbf{x}; \mathbf{w})^T \boldsymbol{\Lambda}(\mathbf{f}) \boldsymbol{J}(\mathbf{x}; \mathbf{w}) \\ &\approx -\boldsymbol{J}(\mathbf{x}; \mathbf{w}) \boldsymbol{\Lambda}(\mathbf{f}) \boldsymbol{J}(\mathbf{x}; \mathbf{w})^T. \end{aligned} \tag{2.4.4}$$

This form guarantees that the Hessian stays positive semi-definite and allows an easy computation since we only need first-order derivatives with respect to the neural network. This approximation assumes that $\boldsymbol{H}(\mathbf{x})^T \nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f}) = 0$. [18] provided two independent sufficient conditions as a justification: i) if the neural network is a perfect predictor, then $\nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f})$ vanishes for all data points $(\mathbf{x}, \mathbf{y})$, but this can indicate overfitting and can thus be unrealistic; ii) the network model is linear, causing the Hessian to vanish. We follow the second alternative as in [19] and formulate a *local linearization* of the network function evaluated at the MAP denoted by $\mathbf{w}_\star$ as follows:

$$\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w}) = \mathbf{f}(\mathbf{x}; \mathbf{w}_\star) + \boldsymbol{J}(\mathbf{x}; \mathbf{w}_\star)(\mathbf{w} - \mathbf{w}_\star).$$

This linearization reduces the Bayesian neural network (BNN) to a Bayesian GLM. The corresponding log joint distribution is given by

$$\ell_{\text{glm}}(\mathbf{w}, \mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{y}_n | \mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}_n; \mathbf{w})) + \log p(\mathbf{w}),$$

where the linearization appears in the parameters and not in the inputs $\mathbf{x}$. The GGN approximation brings two benefits: the Hessian $\boldsymbol{H}$ is guaranteed to be positive semi-definite, and applying this approximation to the Hessian of the likelihood transforms the underlying probabilistic model locally from a BNN into a GLM. The Laplace-GGN approximation jointly applies the Laplace approximation and the GGN approximation [11]; we define the posterior approximation as $q(\mathbf{w}) := \mathcal{N}(\mathbf{w}_\star, \boldsymbol{\Sigma}_{ggn})$, with

$$\boldsymbol{\Sigma}_{\text{ggn}}^{-1} = \sum_{n=1}^{N} \boldsymbol{J}(\mathbf{x}_n; \mathbf{w}_\star)^T \boldsymbol{\Lambda}(\mathbf{f}_n) \boldsymbol{J}(\mathbf{x}_n; \mathbf{w}_\star) + \mathbf{S}_0^{-1}, \tag{2.4.5}$$

where $\mathbf{S}_0$ denotes the prior covariance such that $p(\mathbf{w}) = \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0)$. Therefore by linearizing the neural network we are able to compute the Hessian at $\mathbf{w}_\star$. Interestingly, this formulation allows better local reasoning of the neural network and enables it to be mapped in the feature space using a Gaussian process that allows function space inference.

CHAPTER 3 GENERALIZED GAUSS–NEWTON APPROXIMATION FOR NEURAL NETWORKS

In this chapter we seek to combine the results of the Laplace and generalized Gauss–Newton (GGN) approximations to reveal their implications for neural networks. We are interested in showing how to apply the generalized Gauss-Newton method to the neural network model and analyzing the implications. One major consequence is that the combination gives rise to a generalized linear model due to the linearizing property of GGN, which establishes a connection to Gaussian process models.

3.1 Generalized Linear Model for Neural Networks

We recall that GGN applies a linearization on its parameter vector as follows:

$$\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w}) = \mathbf{f}(\mathbf{x}; \mathbf{w}_\star) + \boldsymbol{J}(\mathbf{x}; \mathbf{w}_\star)(\mathbf{w} - \mathbf{w}_\star).$$

Given a probabilistic neural network model $\mathbf{f}(\mathbf{x}; \mathbf{w}_\star)$, its linearized version $\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w})$ is shifted locally, allowing us to obtain the posterior of a generalized linear model (GLM) [20]. A GLM is a special case of an exponential family distribution in which the natural parameters are a linear function of the inputs. Given a linearized neural network, let us denote the approximate distribution by $\hat{p}_{\mathcal{GL}}(\mathbf{w}|\mathcal{D}) := q_{\mathcal{GL}}(\mathbf{w})$, which takes the form

$$\hat{p}_{\mathcal{GL}}(\mathbf{w}|\mathcal{D}) \propto p(\mathbf{w}) \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}_n; \mathbf{w})), \tag{3.1.1}$$

where $\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}_n; \mathbf{w})$ is linear in the parameter $\mathbf{w}$; consequently, the Jacobian can be represented as a local feature map of the inputs. We can show now that the exact posterior of the GP regression model corresponds to the Laplace–GGN approximation. As a result, applying Laplace approximation to this GLM leads to an exact inference in a Bayesian linear regression model.

Theorem 3.1.1. *For a Bayesian neural network model, the Laplace–GGN approximation is equivalent to the exact posterior of a Bayesian linear regression model. Using Laplace approximation, the approximate distribution corresponds to*

$$q_{\mathcal{GL}}(\mathbf{w}) \propto$$

$$p(\mathbf{w}) \prod_{n=1}^{N} \mathcal{N}\left(\mathbf{y}_n | \mathbf{g}^{-1}(\mathbf{f}(\mathbf{x}_n; \mathbf{w}_\star)) + \boldsymbol{\Lambda}(\mathbf{f}_\star)\boldsymbol{J}(\mathbf{x}_n; \mathbf{w}_\star)(\mathbf{w} - \mathbf{w}_\star), \boldsymbol{\Lambda}(\mathbf{f}_\star)\right).$$

Proof : We use the log density for convenience, and to simplify notation we drop the dependency on $\mathbf{x}$ and $\mathbf{w}$ as follows:

$$\mathbf{f}_\star =: \mathbf{f}(\mathbf{x}; \mathbf{w}_\star), \quad \boldsymbol{J}_\star := \boldsymbol{J}(\mathbf{x}; \mathbf{w}_\star), \quad \boldsymbol{\Lambda}_\star := \boldsymbol{\Lambda}(\mathbf{f}(\mathbf{x}; \mathbf{w}_\star)).$$

First, since the prior remains unchanged, we derive the log-likelihood of the GLM using the second-order Taylor expansion around $\mathbf{w}_\star$,

$$\begin{aligned}
\log p(\mathbf{y}|\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w})) &\approx \log p(\mathbf{y}|\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w}_\star)) + (\mathbf{w} - \mathbf{w}_\star)^T \nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}; \mathbf{w})) \\
&\quad - \frac{1}{2}(\mathbf{w} - \mathbf{w}_\star)^T \nabla^2_{\mathbf{ww}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}; \mathbf{w})(\mathbf{w} - \mathbf{w}_\star) \\
&= \log p(\mathbf{y}|\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w}_\star)) + \left(\mathbf{y} - \mathbf{g}^{-1}(\mathbf{f}_\star)\right) \boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star) \\
&\quad - \frac{1}{2}(\mathbf{w} - \mathbf{w}_\star)^T \boldsymbol{J}_\star^T \boldsymbol{\Lambda}_\star \boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star) \\
&= \log p(\mathbf{y}|\mathbf{f}_\star) - \left(\mathbf{g}^{-1}(\mathbf{f}_\star) - \mathbf{y}\right) \left(\boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star)\right) \\
&\quad - \frac{1}{2} \left(\boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star)\right)^T \boldsymbol{\Lambda}_\star \left(\boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star)\right) \\
&= \log p(\mathbf{y}|\mathbf{f}_\star) + \frac{1}{2} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) - \mathbf{y}\right)^T \boldsymbol{\Lambda}_\star^{-1} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) - \mathbf{y}\right) - \\
&\quad \frac{1}{2} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) + \boldsymbol{\Lambda}_\star \boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star) - \mathbf{y}\right)^T \boldsymbol{\Lambda}_\star^{-1} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) + \boldsymbol{\Lambda}_\star \boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star) - \mathbf{y}\right),
\end{aligned}$$

where in the last step we completed the square and isolated the $\mathbf{w}$-independent terms. Exponentiating this expression gives

$$\begin{aligned}
p(\mathbf{y}|\mathbf{f}_\star) \exp\Bigg\{ &-\frac{1}{2} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) + \boldsymbol{\Lambda}_\star \boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star) - \mathbf{y}\right)^T \boldsymbol{\Lambda}_\star^{-1} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) + \boldsymbol{\Lambda}_\star \boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star) - \mathbf{y}\right) \\
&+ \frac{1}{2} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) - \mathbf{y}\right)^T \boldsymbol{\Lambda}_\star^{-1} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) - \mathbf{y}\right) \Bigg\};
\end{aligned}$$

the last factor in the expression can be resolved by the Gaussian integral, which gives $(2\pi)^{P/2}|\boldsymbol{\Lambda}_\star|^{-1/2}$. Finally, we identify the covariance as $\boldsymbol{\Lambda}_\star$ and the mean as $\mathbf{g}^{-1}(\mathbf{f}_\star) + \boldsymbol{\Lambda}_\star \boldsymbol{J}_\star(\mathbf{w} - \mathbf{w}_\star)$. This result shows when using the Laplace approximation to a linearized model, it becomes a Bayesian linear regression model that matches the moments of the original likelihood at the MAP estimate $\mathbf{w}_\star$. Similarly, one can transform the inference problem from the weight space to the function space; we explain this method in the next section.

3.2 Generalized Gauss–Newton Aproximation in Function Space

For a linearized neural network with a prior $p(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{S}_0)$ we identify a GP prior denoted by $p(\mathbf{f}_{\mathcal{GP}}) \sim \mathcal{GP}(\mathbf{m}(\mathbf{x}), \boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}'))$ and characterized by its mean and the kernel function as follows [4]:

$$\mathbf{m}(\mathbf{x}) = \mathbb{E}_{p(\mathbf{w})}\left[\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w})\right] = \mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \boldsymbol{\mu}_0) \tag{3.2.1}$$

$$\boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}') = Cov_{p(\mathbf{w})}\left[\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w}), \mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}'; \mathbf{w})\right] = \boldsymbol{J}(\mathbf{x}; \mathbf{w}_\star)^T \mathbf{S}_0 \boldsymbol{J}(\mathbf{x}'; \mathbf{w}_\star). \tag{3.2.2}$$

Following [21], we can define the posterior Gaussian process in line with equation (3.1.1) given by

$$\hat{p}_{\mathcal{GL}}(\mathbf{f}_{\mathcal{GP}}|\mathcal{D}) \propto p(\mathbf{f}_{\mathcal{GP}}) \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{f}_{\mathcal{GP}}); \tag{3.2.3}$$

we obtain the same posterior predictive for both the GLM and the GP model when we specify the same prior over functions [4]. To be able to perform inference, we apply the Laplace approximation to the GLM; this formulation is equivalent to solving a Gaussian process regression model. Therefore the generalized linear model can be interpreted as a generalized Gaussian process model due to the GGN approximation in equation (3.2.3). We next want to show that the Laplace approximation to this model can indeed be cast as a Gaussian process regression model. This will allow us to interpret the Laplace–GGN in the function space instead of the parameter space, permitting a trade-off in computational cost between quantity of data points and number of data dimensions. In addition, the function space approximation holds for the posterior, posterior predictive, and marginal likelihood. This method is very well suited to Gaussian process inference for finite-width neural networks [22]. Below we present the connection of the Laplace–GGN to Gaussian processes and develop its key implications. Let us consider a classification problem, denoting the Laplace–GGN posterior approximation in function space by $q(\mathbf{f}_{\mathcal{GP}})$. The linearized neural network with its link function has the following form [4]:

$$\mathbf{g}_{\text{lin}}^{-1}(\mathbf{x}; \mathbf{w}) = \mathbf{g}^{-1}(\mathbf{f}(\mathbf{x}; \mathbf{w}_\star)) + \boldsymbol{\Lambda}(\mathbf{x})\boldsymbol{J}(\mathbf{x})(\mathbf{w} - \mathbf{w}_\star).$$

To derive the approximate distribution in the function space we need to define a GP-prior and apply the Laplace approximation to the GLM model.

Theorem 3.2.1. *Let us consider a GP prior $\hat{p}(\hat{\mathbf{f}}_{\mathcal{GP}})$ with mean and covariance functions*

defined as $\hat{\mathbf{f}}_{\mathcal{GP}} \sim \mathcal{GP}(\hat{\mathbf{m}}(\mathbf{x}), \hat{\boldsymbol{\kappa}}(\mathbf{x}, \mathbf{x}'))$*:*

$$\hat{\mathbf{m}}(\mathbf{x}) = \mathbf{g}_{lin}^{-1}(\mathbf{x}, \boldsymbol{\mu}_0) \quad \textit{and} \quad \hat{\boldsymbol{\kappa}}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\Lambda}(\mathbf{x})\boldsymbol{J}(\mathbf{x})^T \mathbf{S}_0 \boldsymbol{J}(\mathbf{x}')\boldsymbol{\Lambda}(\mathbf{x}').$$

The Laplace approximation to the generalized Gaussian process model of equation (3.2.3) *evaluated at* $\mathbf{f}(\mathbf{x}; \mathbf{w}_\star)$ *corresponds to*

$$q(\mathbf{f}_{\mathcal{GP}}) = \hat{p}(\mathbf{f}_{GP}|\mathcal{D}) \propto \hat{p}(\hat{\mathbf{f}}_{\mathcal{GP}}) \prod_{n=1}^{N} p(\mathbf{y}_n|\hat{\mathbf{f}}_{\mathcal{GP}}, \boldsymbol{\Lambda}(\mathbf{x}_n)).$$

Proof: Since the prior is Gaussian due to its $\mathcal{GP}$ construction, we derive the likelihood of the GLM using the second-order Taylor expansion around $\mathbf{w}_\star$ as follows:

$$\begin{aligned} p(\mathbf{y}|\mathbf{f}_{\mathcal{GP}}) &\approx p(\mathbf{y}|\mathbf{f}_\star) \exp\left\{ \left(\mathbf{g}^{-1}(\mathbf{f}_\star) - \mathbf{y}\right) (\mathbf{f}_{\mathcal{GP}} - \mathbf{f}_\star) - \frac{1}{2} (\mathbf{f}_{\mathcal{GP}} - \mathbf{f}_\star)^T \boldsymbol{\Lambda}_\star (\mathbf{f}_{\mathcal{GP}} - \mathbf{f}_\star) \right\} \\ &= p(\mathbf{y}|\mathbf{f}_\star) \exp\Bigg\{ \frac{1}{2} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) - \mathbf{y}\right)^T \boldsymbol{\Lambda}_\star^{-1} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) - \mathbf{y}\right) \\ &\quad - \frac{1}{2} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) + \boldsymbol{\Lambda}_\star \mathbf{f}_{\mathcal{GP}} - \boldsymbol{\Lambda}_\star \mathbf{f}_\star - \mathbf{y}\right) \boldsymbol{\Lambda}_\star^{-1} \left(\mathbf{g}^{-1}(\mathbf{f}_\star) + \boldsymbol{\Lambda}_\star \mathbf{f}_{\mathcal{GP}} - \boldsymbol{\Lambda} \mathbf{f}_\star - \mathbf{y}\right) \Bigg\}. \end{aligned}$$

This shows that Laplace–GGN approximation in function space enables inference corresponding to a full posterior covariance for neural networks. The computational cost of the kernel inversion and the Jacobian computation is equal to $\mathcal{O}(N^3K^3 + NPK)$. To reduce the computational cost we can resort to low-rank approximation. In the following sections, we derive the posterior and the predictive function of this model and the corresponding marginal likelihood.

3.3 The Laplace–GGN Posterior Approximation

Since the Bayesian linear regression model has a closed mean and covariance form, we can compute the Laplace–GGN approximation to the posterior distribution of the neural network model by using Theorem 3.1.1. We first consider the parameters of the Gaussian approximation $q(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\Sigma})$. Following [10] the Gaussian posterior $\hat{p}(\mathbf{w}|\mathcal{D})$ has the distribution parameters

$$\boldsymbol{\Sigma} = \left(\sum_{n=1}^{N} \boldsymbol{J}(\mathbf{x}_n)^T \boldsymbol{\Lambda}(\mathbf{x}_n) \boldsymbol{J}(\mathbf{x}_n)^T + \mathbf{S}_0^{-1} \right)^{-1} \tag{3.3.1}$$

and

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\sum_{n=1}^{N} \boldsymbol{J}(\mathbf{x}_n)^T\left(\mathbf{y}_n - \mathbf{g}^{-1}(\mathbf{f}(\mathbf{x}_n))\right) + \boldsymbol{J}(\mathbf{x}_n)^T\boldsymbol{\Lambda}(\mathbf{x}_n)\boldsymbol{J}(\mathbf{x}_n)\mathbf{w}_\star + \mathbf{S}_0^{-1}\boldsymbol{\mu}_0\right), \tag{3.3.2}$$

where $\boldsymbol{\Lambda}(\mathbf{x}_n) := \boldsymbol{\Lambda}(\mathbf{f}(\mathbf{x}_n; \mathbf{w}))$ and we recognise the covariance as the inverse of the negative log-joint distribution. In particular, for the mean, when a local minimum exists, from equation (3.3.2) using the optimal conditions we have

$$\sum_{n=1}^{N} \boldsymbol{J}(\mathbf{x}_n)^T\left(\mathbf{y}_n - \mathbf{g}^{-1}(\mathbf{f}(\mathbf{x}_n)\right) - \mathbf{S}_0^{-1}\left(\mathbf{w}_\star - \boldsymbol{\mu}_0\right) = 0,$$

which yields $\boldsymbol{\mu} = \mathbf{w}_\star$ since it is a minimizer of the Laplace approximation constructed at the MAP. We are now able to cast the posterior of the generalized Gaussian process in the function space, so we can use the results of equations (3.2.1) and (3.2.2). When the GLM and the GP coincide with $\mathbf{w}_\star = \mathbf{w}_{\text{MAP}}$, we can construct the Laplace approximation at $\mathbf{f}_{\text{lin}}(\mathbf{x}; \mathbf{w}_\star)$. We choose an isotropic prior on the parameters with covariance $\mathbf{S}_0 = \delta^{-1}\mathbf{I}_P$ for a scalar $\delta > 0$ and zero mean $\mathbf{m}_0 = 0$. In the case of K multi-output predictions, the kernel maps to a block diagonal covariance matrix due to the independence of the GPs for each output and now has a size of $NK \times NK$ instead of $N \times N$. We denote the block diagonal matrix $\mathbf{L}_{\mathbf{XX}} \in \mathbb{R}^{NK \times NK}$ defined by $N(K \times K)$ blocks where the n-th block is the negative log-likelihood Hessian $\boldsymbol{\Lambda}(\mathbf{f}(\mathbf{x}_n; \mathbf{w}_\star))$. Following [4], we have $\nabla^2_{\mathbf{ff}} \log p(\mathcal{D}, \mathbf{f}(\mathbf{X})) = -\mathbf{L}_{\mathbf{XX}} - \delta\left(\boldsymbol{J}(\mathbf{X}; \mathbf{w}_\star)\boldsymbol{J}(\mathbf{X}; \mathbf{w}_\star)^T\right)^{-1}$; for a test input $\mathbf{x}_\star$ the approximate posterior takes the form

$$q(\mathbf{f}_\star|\mathbf{x}_\star, \mathcal{D}) = \mathcal{N}\left(\mathbf{f}(\mathbf{x}_\star; \mathbf{w}_\star), \mathbf{K}_{\mathbf{x}_\star\mathbf{x}_\star} - \mathbf{K}_{\mathbf{x}_\star\mathbf{X}}\left(\mathbf{K}_{\mathbf{XX}} + \mathbf{L}_{\mathbf{XX}}^{-1}\right)^{-1}\mathbf{K}_{\mathbf{X}\mathbf{x}_\star}^T\right),$$

where $\mathbf{K}_{\mathbf{x}_\star\mathbf{X}}$ is the kernel between the input test locations and the N training examples. Since we have applied the GGN approximation, the kernel is given by $\mathbf{K}_{\mathbf{x}_\star\mathbf{X}} = \delta^{-1}\boldsymbol{J}(\mathbf{x}_\star; \mathbf{w}_\star)\boldsymbol{J}(\mathbf{X}; \mathbf{w}_\star)$. Interestingly, [23] assume independent prior GPs for each output, yielding $p(\mathbf{f}_{\mathcal{GP}}) = \prod_{k=1}^{K} p(\mathbf{f}_{\mathcal{GP}})_k$; this independence property speeds up the computation and reduces the computational cost of GPs.

3.4 Approximate Posterior Predictive

In this section we highlight how the GGN approximation method impacts the posterior predictive. The first case, *naive BNN*, is simply a neural network based only on the MAP estimate and obtained by Monte Carlo sampling. For the second case, we apply the GGN method, which changes the original model to a generalized linear or Gaussian process model.

In the subcase of regression, the Laplace approximation gives rise to exact inference in a Bayesian linear or Gaussian process regression model. In the classification subcase, we resort to approximate techniques. We illustrate the approximate posterior predictives for the different models as follows:

- Predictive naive BNN

  To make a prediction with a naive neural network, we need to approximate the posterior predictive integral by sampling S Monte Carlo samples with $\mathbf{w}_1, \ldots, \mathbf{w}_S \sim q(\mathbf{w})$. For a new test input $\mathbf{x}_\star$ we have

$$\hat{p}_{\text{mc}}(\mathbf{y}_\star|\mathcal{D}, \mathbf{x}_\star) \approx \int p(\mathbf{y}_\star|\mathbf{f}(\mathbf{x}_\star; \mathbf{w}))q(\mathbf{w})d\mathbf{w} \approx \frac{1}{S}\sum_{i=1}^{S} p(\mathbf{y}_\star|\mathbf{f}(\mathbf{x}_\star; \mathbf{w}_i)).$$

- Predictive GLM

  Using a *linearized neural network*, the Laplace approximation to the posterior $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}_{\text{MAP}}, \boldsymbol{\Sigma})$ results in a Gaussian distribution on the neural network outputs $\mathbf{f}(\mathbf{x}_\star)$ and takes the form

$$\hat{p}(\mathbf{f}_\star|\mathbf{x}_\star, \mathcal{D}) \approx \mathcal{N}(\mathbf{f}_\star; \mathbf{f}(\mathbf{w}_\star; \mathbf{x}_\star), \boldsymbol{J}(\mathbf{x}_\star)^T \boldsymbol{\Sigma} \boldsymbol{J}(\mathbf{x}_\star));$$

  this formulation is efficient since it has a lower dimension (number of outputs K instead of parameters P) [24]. The predictive distribution can be obtained by integrating the likelihood as follows:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}_\star, \mathcal{D}) &= \int p(\mathbf{y}|\mathbf{f}_\star)p(\mathbf{f}_\star|\mathbf{x}_\star, \mathcal{D})d\mathbf{w} \\ &\approx \mathcal{N}(\mathbf{f}_\star; \mathbf{f}(\mathbf{w}_\star; \mathbf{x}_\star), \boldsymbol{J}(\mathbf{x}_\star)^T \boldsymbol{\Sigma} \boldsymbol{J}(\mathbf{x}_\star) + \sigma^2 \mathbf{I}), \end{aligned}$$

  where σ^2 is the variance of the Gaussian likelihood.

  In the case where the likelihood is not Gaussian (e.g., classification), we need further approximation. That is, we want to compute the following intractable integral:

$$p(\mathbf{y}_\star|\mathbf{x}_\star, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{f}(\mathbf{x}_\star; \mathbf{w}_\star))\mathcal{N}(\mathbf{f}_\star|\boldsymbol{\mu}_\star, \boldsymbol{\Sigma}_\star)d\mathbf{f}_\star.$$

  GLMs enable generalization of loss functions, with the inverse link function $\mathbf{g}^{-1}(\mathbf{f}_\star)$ chosen as the sigmoid function $\sigma(\mathbf{f}_\star)$ or as a softmax function. [25] proposed to approximate the sigmoid function σ with the probit function $\Phi-$the standard Normal

cumulative distribution function, which provides a closed-form expression of the predictive. In practice, the GLM sampling method consists of using a second-order Taylor expansion of the sigmoid function around the mean. Given a Gaussian $p(\mathbf{f}_\star|\mathbf{x}_\star, \mathcal{D})$, we can easily compute the integral as $\mathbb{E}_{p(\mathbf{f}_\star|\mathbf{x}_\star,\mathcal{D})}[\sigma(\mathbf{f}_\star)]$. However, we *can* obtain a closed-form expression when the inverse link function $\mathbf{g}^{-1}(\mathbf{f}_\star)$ corresponds to a log probability distribution. That is, assume the loss $\ell(\mathbf{y}, \mathbf{f}(\mathbf{x}; \mathbf{w})) := -\log p(\mathbf{y}|\mathbf{g}^{-1}(\mathbf{x}; \mathbf{w}))$, where $\mathbf{g}^{-1}(.)$ is the *inverse link function* and $\mathbf{y} \in \{0, 1\}$ follows a Bernoulli distribution. The Bernoulli likelihood function is $\sigma(\mathbf{f}(\mathbf{x}; \mathbf{w}_\star))$, where σ is the sigmoid function and the noise precision corresponds to $\mathbf{\Lambda(f_\star)} := \sigma(\mathbf{f}(\mathbf{x}; \mathbf{w}_\star))\,(1 - \sigma(\mathbf{f}(\mathbf{x}; \mathbf{w}_\star)))$. Therefore given $\mathbf{g}^{-1}_{\text{lin}}(\mathbf{f}_\star)$, which is linear in the parameter $\mathbf{w}$, then the predictive distribution is given by

$$\mathcal{N}\left(\mathbf{y}|\mathbf{g}^{-1}_{\text{lin}}(\mathbf{x}_\star), \mathbf{\Lambda}(\mathbf{f}_\star)\boldsymbol{J}(\mathbf{x}_\star)\mathbf{\Sigma}\boldsymbol{J}(\mathbf{x}_\star)^T\mathbf{\Lambda}(\mathbf{f}_\star) + \mathbf{\Lambda}(\mathbf{f}_\star)\right).$$

Computing different predictive posteriors will enable a better choice of model; in the experiments chapter we will show how these different models vary in term of accuracy. More interestingly, we will show that the naive Bayesian neural network with MAP estimate suffers from underfitting, while in the GLM case this problem could be reduced substantially. In the next section we turn our attention to model selection by computing the marginal likelihood.

3.5 The Marginal Likelihood Approximation and Hyperparameter Tuning

In the deep learning literature, cross-validation is generally used to tune hyperparameters; in Bayesian model selection, however, we use the marginal likelihood to tune these hyperparameters in the training phase, an adaptation known as *empirical Bayes* or *type-II-maximum-likelihood* learning [4] which is closely related to *Occam's razor* [10]. This procedure is rarely performed in deep learning due to the computational challenges. Using this technique with variational inference, [26] found that optimizing the hyperparameters does not always give good results and it is challenging to estimate. In this section, we are interested in deriving the marginal likelihood using the Laplace method with Gauss–Newton approximation, showing how we can identify a closed form of the marginal likelihood. We will use these results in the experiments chapter for model selection. In particular, we demonstrate how the marginal likelihood approximation could calibrate the given model, improving model performance. We therefore derive the Laplace–GGN marginal likelihood approximation by using the exact marginal likelihood given the Bayesian linear regression or GP regression models. First, we use the result from Laplace approximation to derive the marginal likelihood of the GLM.

Theorem 3.5.1. *Following Theorems 3.1.1 and 3.2.1, we can derive the marginal likelihood*

of the Bayesian linear or Gaussian process regression model denoted by $\hat{p}(\mathcal{D})$ for a given model $\mathcal{M}$, where $\mathcal{M}$ encodes the neural network architecture, hyperparameter priors, and so on. We denote by $q_{\mathcal{GL}}(\mathcal{D})$ the Laplace approximation to the marginal likelihood of the generalized linear model, which takes the following form:

$$\log q_{\mathcal{GL}}(\mathcal{D}|\mathcal{M}) = \sum_{n=1}^{N}\Big[\log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n;\mathbf{w}_\star)) - \log\mathcal{N}\left(\mathbf{y}_n|\mathbf{g}^{-1}(\mathbf{f}(\mathbf{x}_n;\mathbf{w}_\star)),\mathbf{\Lambda}(\mathbf{f}_n)\right)\Big] + \log\hat{p}(\mathcal{D}|\mathcal{M}). \tag{3.5.1}$$

Proof : To derive the marginal likelihood of the GLM or the GP model, we make use of the second-order Taylor approximation around $\mathbf{w}_\star$ to the log-joint distribution as follows:

$$\begin{aligned}
\log q_{\mathcal{GL}}(\mathcal{D}|\mathcal{M}) &= \int \log p(\mathcal{D},\mathbf{w}|\mathcal{M})d\mathbf{w} \\
&\approx \int\Bigg(\log p(\mathcal{D},\mathbf{w}|\mathcal{M}) - (\mathbf{w}-\mathbf{w}_\star)^T\nabla_{\mathbf{w}}\log p(\mathcal{D},\mathbf{w}|\mathcal{M}) \\
&\quad - \frac{1}{2}(\mathbf{w}-\mathbf{w}_\star)^T\nabla^2_{\mathbf{ww}}\log p(\mathcal{D},\mathbf{w}|\mathcal{M})(\mathbf{w}-\mathbf{w}_\star)\Bigg)d\mathbf{w} \\
&= \frac{1}{2}\left(\mathbf{g}^{-1}(\mathbf{f}(\mathbf{x};\mathbf{w}_\star)) - \mathbf{y}\right)^T\mathbf{\Lambda}^{-1}\left(\mathbf{g}^{-1}(\mathbf{f}(\mathbf{x};\mathbf{w}_\star)) - \mathbf{y}\right) - \frac{1}{2}\Bigg(\mathbf{g}^{-1}(\mathbf{f}(\mathbf{x};\mathbf{w}_\star)) \\
&\quad + \mathbf{\Lambda}\boldsymbol{J}(\mathbf{w}-\mathbf{w}_\star) - \mathbf{y}\Bigg)^T\mathbf{\Lambda}^{-1}\left(\mathbf{g}^{-1}(\mathbf{f}(\mathbf{x};\mathbf{w}_\star)) + \mathbf{\Lambda}\boldsymbol{J}(\mathbf{w}-\mathbf{w}_\star) - \mathbf{y}\right) \\
&\quad + \log p(\mathbf{w}_\star|\mathcal{M}) + \log p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{w}_\star)) \\
&= \sum_{n=1}^{N}\left[\log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n;\mathbf{w}_\star)) - \log\mathcal{N}\left(\mathbf{y}_n|\mathbf{g}^{-1}(\mathbf{f}(\mathbf{x}_n;\mathbf{w}_\star)),\mathbf{\Lambda}(\mathbf{f}_n)\right)\right] \\
&\quad + \log\hat{p}(\mathcal{D}|\mathcal{M}),
\end{aligned}$$

where in the third line we add and subtract $\frac{1}{2}\log(2\pi)^k|\mathbf{\Lambda}|$ to match the Gaussian likelihood and collect the terms with no dependency on $\mathbf{w}_\star$. Then, we combine the second term and the prior to obtain the marginal likelihood of the Bayesian linear regression model or the GP model. The marginal likelihood of the GLM or GP depends on the marginal likelihood of the linear or GP regression model. When the neural network likelihood is Gaussian this leads to $\hat{p}(\mathcal{D})$; otherwise we need to add a correction term. The marginal density for a given model $\mathcal{M}$ takes the form

$$\log\hat{p}(\mathcal{D}|\mathcal{M}) = \sum_{n=1}^{N}\log\mathcal{N}(\mathbf{y}_n|\mathbf{g}^{-1}(\mathbf{f}(\mathbf{x}_n)) - \frac{1}{2}\log\frac{|\mathbf{S}_0|}{|\mathbf{\Sigma}|} - \frac{1}{2}\left(\boldsymbol{\mu}-\boldsymbol{\mu}_0\right)^T\mathbf{S}_0^{-1}\left(\boldsymbol{\mu}-\boldsymbol{\mu}_0\right).$$

In function space, the kernel is composed of $N^2(K \times K)$ sub-matrices with $\mathbf{K} \in \mathbb{R}^{NK \times NK}$ and the block diagonal matrix of the N Hessians $\mathbf{\Lambda} \in \mathbb{R}^{NK \times NK}$. Following [4], the marginal likelihood of the GP regression model is given by

$$\log q_{\mathcal{GP}}(\mathcal{D}) = -\frac{1}{2}(\mathbf{m} - \mathbf{y})^T(\mathbf{K} + \mathbf{\Lambda})^{-1}(\mathbf{m} - \mathbf{y}) - \frac{1}{2}\left|\mathbf{K} + \mathbf{\Lambda}\right| + \text{constant},$$

where the mean function $\mathbf{m} \in \mathbb{R}^{NK}$ comprises the individual mean functions of the N data points and $\mathbf{y}$ is the concatenation of the labels.

3.6 KFAC Approximation of the Hessian

We can proceed to a further approximation of the GGN Hessian to enable fast computation and storage for large datasets. [27] proposed the Laplace approximation given by Kronecker-factored approximate curvature (KFAC), which uses a special posterior structure to combine a Gaussian prior with the Hessian approximation. The approximation uses Kronecker factors, comprising block-diagonal matrices while maintaining greater expressiveness than diagonal approximation. That is, each block corresponds to a neural network termed as a parameter group; for the l-th parameter group of the GGN, the approximation takes the form

$$[\boldsymbol{H}_{GGN}]_l = \sum_{n=1}^{N} \left[\boldsymbol{J}(\mathbf{x}_n)^T \mathbf{\Lambda}(\mathbf{y}_n; \mathbf{f}_n) \boldsymbol{J}(\mathbf{x}_n)\right]_l \approx \boldsymbol{Q}_l \otimes \boldsymbol{W}_l$$

where $\boldsymbol{Q}_l$ denotes the covariance of the activation and is quadratic in the number of neurons of the l-th layer while $\boldsymbol{W}_l$ denotes the output of the layer l and is quadratic in the number of neurons of the previous layer. Both $\boldsymbol{Q}_l$ and $\boldsymbol{W}_l$ are positive semi-definite, which enables fast inversion since the inversion is done individually.

CHAPTER 4 VARIATIONAL INFERENCE WITH GGN

Variational inference (VI), often referred to as Gaussian variational approximation, provides similar results to Laplace approximation since they are equally scalable and use the same approximating family. However, the optimization step is different for VI because it is not sequential: the algorithm maximizes the evidence lower bound (ELBO) as described in equation (2.2.1). Desirable properties of the generalized Gauss–Newton method induce us to combine it with VI, leading to the identification of a new algorithm. Thus, using GGN provides a better understanding of approximate inference learning for each step of the VI algorithm. In fact, with the Gaussian process formulation we are able to suggest posterior predictive distributions and updates in function space. To maximize the ELBO, we make use of natural-gradient variational inference (NGVI). This method, used principally in the field of Bayesian deep learning [11; 28], exploits the information geometry to improve convergence [29]. In this chapter we specify the parameter updates of an NGVI method and introduce the variational online generalized Gauss–Newton (VOGGN) algorithm, which is derived from the NGVI updates to a Gaussian posterior approximation.

4.1 Natural-Gradient Variational Inference

Natural-gradient variational inference is mathematically convenient and very efficient because the parameters are updated sequentially. The natural-gradient method, often termed natural-gradient descent, uses a very different approach than classical gradient descent. While the classical approach uses Euclidean geometry parameter updates, natural-gradient descent uses information geometry, the updates in our case being applied to the parameters of the Gaussian posterior approximation, so that they are performed in the distribution space. In this section we derive the natural variational inference in the natural parameter space. Therefore we need to define natural and expectation parametrization of the Gaussian posterior approximation. Denoting the first natural parameters by $\{\boldsymbol{\nu}^{(1)}, \boldsymbol{\nu}^{(2)}\}$ and the mean parameters by $\{\boldsymbol{\phi}^{(1)}, \boldsymbol{\phi}^{(2)}\}$, we have

$$\boldsymbol{\nu}^{(1)} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \quad \text{and} \quad \boldsymbol{\nu}^{(2)} = -\frac{1}{2}\boldsymbol{\Sigma}^{-1}, \tag{4.1.1}$$

$$\boldsymbol{\phi}^{(1)} = \boldsymbol{\mu} \quad \text{and} \quad \boldsymbol{\phi}^{(2)} = \boldsymbol{\mu}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}. \tag{4.1.2}$$

Let $\mathbf{F}(\boldsymbol{\nu})$ be the Fisher information matrix of the approximate distribution $q(\mathbf{w})$ and $\mathcal{L}$ the loss function commonly termed the ELBO. Following [16], we can express the update rule of

the NGVI in natural space as follows:

$$\begin{aligned}\boldsymbol{\nu}_{t+1} &= \boldsymbol{\nu}_t + \beta \mathbf{F}(\boldsymbol{\nu})^{-1} \nabla_\nu \mathcal{L}(\boldsymbol{\nu}) \\ &= \boldsymbol{\nu}_t + \beta Cov_{q(\mathbf{w})} \left[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{w})\right]^{-1} \nabla_\nu \mathcal{L}(\boldsymbol{\nu}) \\ &= \boldsymbol{\nu}_t + \beta \nabla_\phi \mathcal{L}(\boldsymbol{\phi}),\end{aligned} \tag{4.1.3}$$

where β is a step size and the ELBO is parametrized in natural rather than Gaussian parameters. This formulation enables us to compute the gradients with respect to expectations and natural parameters. Interestingly, we are able to update the natural parameters without computing the Fisher information matrix, thereby reducing the computational cost. In a second step, we establish the connection with our Gaussian parameter, making use of the chain rule to express the expectation parameter gradients in terms of gradients with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ such that $\boldsymbol{\mu} = \boldsymbol{\phi}^{(1)}$ and $\boldsymbol{\Sigma} = \boldsymbol{\phi}^{(2)} - \boldsymbol{\phi}^{(1)^2}$. Indeed, applying the chain rule to the natural gradient with respect to the Gaussian parameters gives

$$\begin{aligned}\nabla_{\boldsymbol{\phi}^{(1)}} \mathcal{L}(\boldsymbol{\phi}^{(1)}, \boldsymbol{\phi}^{(2)}) &= \nabla_{\boldsymbol{\mu}} \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) - 2\nabla_{\boldsymbol{\Sigma}} \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \boldsymbol{\phi}^{(1)} \\ &= \mathbb{E}_{q(\mathbf{w})} \left[\log p(\mathcal{D}|\mathbf{w})\right] + \mathbf{S}_0^{-1} \boldsymbol{\mu}_0 - \mathbf{S}_0^{-1} \boldsymbol{\mu}\end{aligned} \tag{4.1.4}$$

and

$$\begin{aligned}\nabla_{\boldsymbol{\phi}^{(2)}} \mathcal{L}(\boldsymbol{\phi}^{(1)}, \boldsymbol{\phi}^{(2)}) &= \nabla_{\boldsymbol{\Sigma}} \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &= \nabla_{\boldsymbol{\Sigma}} \mathbb{E}_{q(\mathbf{w})} \left[\log p(\mathcal{D}|\mathbf{w})\right] + \frac{1}{2}\boldsymbol{\Sigma}^{-1} - \frac{1}{2}\mathbf{S}_0^{-1},\end{aligned} \tag{4.1.5}$$

where we have used the closed-form derivatives of the KL-divergence between the prior and its approximate distribution as proposed by [12; 13] and defined in equations (2.3.1) and (2.3.2). Finally, plugging equations (4.1.4) and (4.1.5) into (4.1.3) we obtain the NGVI with natural parameter-space updates. For a Gaussian variational approximation with the posterior at iteration t denoted by $q_t(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, we have

$$\begin{aligned}\boldsymbol{\Sigma}_{t+1}^{-1} \boldsymbol{\mu}_{t+1} = \beta \Big[&\nabla_{\boldsymbol{\mu}} \mathbb{E} \left[\log p(\mathcal{D}|\mathbf{w})\right] - 2\nabla_{\boldsymbol{\Sigma}} \mathbb{E} \left[\log p(\mathcal{D}|\mathbf{w})\right] \boldsymbol{\mu}_t \\ &+ \mathbf{S}_0^{-1} \boldsymbol{\mu}_0 - \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\mu}_t \Big] + \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\mu}_t\end{aligned}$$

and

$$-\frac{1}{2}\boldsymbol{\Sigma}_{t+1}^{-1} = \beta \left[\nabla_{\boldsymbol{\Sigma}} \mathbb{E} \left[\log p(\mathcal{D}|\mathbf{w})\right] + \frac{1}{2}\boldsymbol{\Sigma}_t^{-1} - \frac{1}{2}\mathbf{S}_0^{-1} \right] - \frac{1}{2}\boldsymbol{\Sigma}_t^{-1}.$$

Therefore, after rearranging the terms, the updates of the first and second natural parameters are given by

$$\begin{aligned}\boldsymbol{\Sigma}_{t+1}^{-1}\boldsymbol{\mu}_{t+1} = \quad & \beta\Big[\nabla_{\boldsymbol{\mu}}\mathbb{E}\left[\log p(\mathcal{D}|\mathbf{w})\right] - 2\nabla_{\boldsymbol{\Sigma}}\mathbb{E}\left[\log p(\mathcal{D}|\mathbf{w})\right]\boldsymbol{\mu}_t\Big] + \\ & (1-\beta)\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t + \beta\mathbf{S}_0^{-1}\boldsymbol{\mu}_0,\end{aligned} \tag{4.1.6}$$

and

$$-\frac{1}{2}\boldsymbol{\Sigma}_{t+1}^{-1} = \beta\nabla_{\boldsymbol{\Sigma}}\mathbb{E}\left[\log p(\mathcal{D}|\mathbf{w})\right] + (1-\beta)\left[-\frac{1}{2}\boldsymbol{\Sigma}_t^{-1}\right] + \left[-\frac{1}{2}\mathbf{S}_0^{-1}\right], \tag{4.1.7}$$

where we usually choose $\beta < 1$ to force a convex combination of the current posterior approximation q_t and the prior. The updates are performed through expectation over the posterior approximation q_t at iteration t. Using the linearity property of the expectation, computation of the gradient of the log-likelihood becomes highly simplified. The data dependency (reliant on the log-likelihood expression) is only due to the gradients with respect to the mean and the covariance of the expected log-likelihood terms. Different algorithms use these NGVI updates: the only significant differences are how estimation of the derivatives is processed for $\nabla_{\boldsymbol{\mu}}\mathbb{E}\left[\log p(\mathcal{D}|\mathbf{w})\right]$ and $\nabla_{\boldsymbol{\Sigma}}\mathbb{E}\left[\log p(\mathcal{D}|\mathbf{w})\right]$. [14] proposed estimating the derivative with respect to the mean and covariance by sampling individual gradients, as we presented in equation (2.3.1) and (2.3.2). In particular, consider the neural network function denoted by $\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x};\mathbf{w})$ where we apply the GGN method, linearized at $\mathbf{w}_\star$, and S denotes the Monte Carlo samples from the approximate distribution $q_t(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ with $\mathbf{w}_1, \ldots, \mathbf{w}_S$; we obtain the gradient samples

$$\nabla_{\boldsymbol{\mu}}\mathbb{E}_{q(\mathbf{w}_s)}\left[\log p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{w}_s))\right] = \mathbb{E}_{q(\mathbf{w}_s)}\left[\nabla_{\mathbf{w}}\log p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{w}_s))\right] \tag{4.1.8}$$

and

$$\nabla_{\boldsymbol{\Sigma}}\mathbb{E}_{q(\mathbf{w}_s)}\left[\log p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{w}_s))\right] = \frac{1}{2}\mathbb{E}_{q(\mathbf{w}_s)}\left[\nabla_{\mathbf{ww}}^2\log p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{w}_s))\right]. \tag{4.1.9}$$

We now derive a natural-gradient variational inference algorithm, the variational online generalized Gauss–Newton (VOGGN) algorithm, presented below. This algorithm first samples the parameters from the approximating distribution and then applies the GGN approximation. For the first derivative, we use S Monte Carlo samples to approximate the expected

gradient:

$$\mathbb{E}_{q(\mathbf{w}_s)}\left[\nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{w}_s))\right] \approx \frac{1}{S}\sum_{i=1}^{S} \boldsymbol{J}(\mathbf{x};\mathbf{w}_i)^T \nabla_{\mathbf{f}} \log p(\mathbf{y}|\mathbf{f}).$$

For the Hessian, we need to sample S neural network models and then linearize these models individually with respect to $\mathbf{w}_s$, as follows:

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{w}_s)}\left[\nabla^2_{\mathbf{ww}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{w}_s))\right] &\approx \frac{1}{2}\sum_{i=1}^{S} \nabla^2_{\mathbf{ww}} p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{w}_i) \\
&\approx \frac{1}{2}\sum_{i=1}^{S} \nabla^2_{\mathbf{ww}} p(\mathbf{y}|\mathbf{f}^{\mathbf{w}_\star}_{\text{lin}}(\mathbf{x};\mathbf{w}_i) \\
&= -\frac{1}{2}\sum_{i=1}^{S} \boldsymbol{J}(\mathbf{x};\mathbf{w}_i)^T \boldsymbol{\Lambda}(\mathbf{f}(\mathbf{w}_i)) \boldsymbol{J}(\mathbf{x};\mathbf{w}_i).
\end{aligned}$$

This form of approximation can therefore be applied to the variational inference algorithm to solve the scalability issue. In the next section, to show how the VOGGN algorithm can solve the Bayesian linear regression problem, we describe a suitable approximation to large-scale data that can be adapted to natural-gradient algorithms.

4.2 Bayesian Linear Regression with VOGGN Algorithm

In this section we want to make use of the natural gradient to solve Bayesian linear regression through Monte Carlo sampling using the VOGGN algorithm [24]. In line with equations (4.1.6) and (4.1.7), we can obtain an approximate update at each iteration of $q_{t+1}(\mathbf{w})$ by linking the prior and the corresponding posterior approximation at iteration t. The key idea is to parametrize the natural-parameter vector of the Gaussian as in equation (4.1.1), as follows:

$$\begin{aligned}
\boldsymbol{\Sigma}^{-1}_{t+1}\boldsymbol{\mu}_{t+1} &= \beta_t \mathbf{S}_0^{-1}\boldsymbol{\mu}_0 + (1-\beta_t)\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t, \\
-\frac{1}{2}\boldsymbol{\Sigma}^{-1}_{t+1} &= -\frac{1}{2}\beta_t\left[\mathbf{S}_0^{-1}\right] - \frac{1}{2}(1-\beta)\left[\boldsymbol{\Sigma}_t^{-1}\right],
\end{aligned}$$

where β is a step size. We identify the natural parameters of $p(\mathbf{w})$ and $q(\mathbf{w})$ at each iteration t, which can be determined by the expected log-likelihood. Now we make use of the exponential family, denoting the natural parameter by $\hat{\nu}$ and the sufficient statistic by $T(\mathbf{w})$.

The posterior approximation at each iteration t takes the update

$$q_{t+1}(\mathbf{w}) \propto q_t(\mathbf{w})^{(1-\beta)} p(\mathbf{w})^{\beta} \exp\Big\{\beta T(\mathbf{w})\hat{\nu}\Big\},$$

where the gradients are updated in the natural parameter $\hat{\nu}$, so that the family of the posterior update depends on the exponential family term given by [24]:

$$\begin{aligned}
\exp\Big\{\beta T(\mathbf{w})\hat{\nu}\Big\} &= \exp\Big\{\beta\langle \mathbf{w}, \nabla_{\phi^{(1)}}\mathcal{L}(\phi^{(1)}, \phi^{(2)})\rangle + \beta\langle \mathbf{w}\mathbf{w}^T, \nabla_{\phi^{(2)}}\mathcal{L}(\phi^{(1)}, \phi^{(2)})\rangle\Big\} \\
&= \exp\Big\{\beta\langle \mathbf{w}, \nabla_{\boldsymbol{\mu}}\mathbb{E}_{q(\mathbf{w})}\left[\log p(\mathcal{D}|\mathbf{w})\right] \\
&\quad - 2\nabla_{\boldsymbol{\Sigma}}\mathbb{E}\left[\log p(\mathcal{D}|\mathbf{w})\boldsymbol{\mu}_t\right]\rangle + \beta\langle \mathbf{w}\mathbf{w}^T, \nabla_{\boldsymbol{\Sigma}}\mathbb{E}\left[\log p(\mathcal{D}|\mathbf{w})\right]\rangle\Big\}.
\end{aligned}$$

Using the linear property of expectation and the inner product, we can therefore use the product over N data points as follows:

$$\begin{aligned}
&\prod_{n=1}^{N} \exp\Big\{\beta\langle \mathbf{w}, \nabla_{\boldsymbol{\mu}}\mathbb{E}_{q(\mathbf{w})}\left[\log p(\mathbf{y}_n|\mathbf{w})\right] \\
&\quad - 2\nabla_{\boldsymbol{\Sigma}}\mathbb{E}\left[\log p(\mathbf{y}_n|\mathbf{w})\boldsymbol{\mu}_t\right]\rangle + \beta\langle \mathbf{w}\mathbf{w}^T, \nabla_{\boldsymbol{\Sigma}}\mathbb{E}\left[\log p(\mathbf{y}_n|\mathbf{w})\right]\rangle\Big\},
\end{aligned}$$

using S Monte Carlo samples $\{\mathbf{w}\}_{i=1}^{S} \sim q_t(\mathbf{w})$ to evaluate the mean over the S samples. Similarly, since the sum is a linear operator we can take the product over these S samples, given by

$$\prod_{n=1}^{N}\prod_{i=1}^{S} \exp\Big\{\frac{\beta}{S}\langle \mathbf{w}, \nabla_{\mathbf{w}} \log p(\mathbf{y}_n|\mathbf{w}_i)\boldsymbol{\mu}_t\rangle + \frac{\beta}{2S}\langle \mathbf{w}\mathbf{w}^T, \nabla^2_{\mathbf{ww}} \log p(\mathbf{y}_n|\mathbf{w}_i)\rangle\Big\}.$$

For a single sample we have

$$\begin{aligned}
&\exp\Big\{\frac{\beta}{S}\langle \mathbf{w}, \boldsymbol{J}_i(\mathbf{x}, \mathbf{w})^T \nabla_{\hat{\mathbf{f}}} \log p(\mathbf{y}|\hat{\mathbf{f}}_i) + \boldsymbol{J}_i(\mathbf{x}, \mathbf{w})^T \boldsymbol{\Lambda}_i(\hat{\mathbf{f}}_i(\mathbf{x}; \mathbf{w}))\boldsymbol{J}_i(\mathbf{x}, \mathbf{w})\boldsymbol{\mu}\rangle \\
&\quad - \frac{\beta}{2S}\langle \mathbf{w}^T\mathbf{w}, \boldsymbol{J}_i(\mathbf{x}; \mathbf{w})^T \boldsymbol{\Lambda}_i(\hat{\mathbf{f}}_i(\mathbf{x}; \mathbf{w}))\boldsymbol{J}_i(\mathbf{x}; \mathbf{w})\rangle\Big\},
\end{aligned}$$

where $\hat{\mathbf{f}}_i(\mathbf{x}; \mathbf{w})$ denotes the function values for the i-th sample. We have used the linear property of the inner product and then completed the square to obtain a Gaussian form. We

finally express each term as a Gaussian distribution product as follows:

$$q_{t+1}(\mathbf{w}) \propto \mathcal{N}(\mathbf{w}|\mathbf{m}_t, \mathbf{S}_t) \prod_{n=1}^{N} \prod_{i=1}^{S} \mathcal{N}\Bigg(\mathbf{y}_n|\mathbf{g}^{-1}(\hat{\mathbf{f}}_i(\mathbf{x}_n;\mathbf{w}_i)) + \mathbf{\Lambda}_i\hat{\mathbf{f}}_i(\mathbf{x}_n;\mathbf{w}_i)\hat{\boldsymbol{J}}_i(\mathbf{x}_n;\mathbf{w}_i)(\mathbf{w}_i - \boldsymbol{\mu}_t), \frac{S}{\beta}\mathbf{\Lambda}_i\hat{\mathbf{f}}_i(\mathbf{x}_n;\mathbf{w}_i)\Bigg). \tag{4.2.1}$$

The VOGGN is characterised by a variational update at each iteration instead of relying solely on stationary point. In particular, when the step size $\beta = 1$ and the Monte Carlo sample is $i = 1$ we recover the Laplace–GGN approximation. In summary, VOGGN provides a more flexible posterior approximation at each iteration. All algorithms for variational inference use the same formalism, varying only in their approximation to these expectations and the log likelihood. Other algorithms have been proposed in the same spirit using low-rank approximation to the Hessian [26; 30]. Interestingly, Kronecker-factored approximation to the Hessian, proposed by [27], enables us to work with more flexible covariance structures with lower complexity.

CHAPTER 5 EXPERIMENTS

In this chapter, we investigate the Gaussian variational approximation and the generalized Gauss–Newton method experimentally to explore how they impact the posterior predictive. We also use both the GLM and the Gaussian process model to approximate the marginal likelihood of a neural network. Finally, we discuss the implications of these approaches for uncertainty quantification and out-of-distribution detection. We conduct our experiments on toy and real datasets for both the regression and classification tasks.

5.1 Bayesian-Model Selection with Marginal Likelihood

Figure 5.1 shows the marginal likelihood for different hyperparameters in the regression case with the *Wine-dataset* from UCI Machine Learning Repository. When we have a weak regularizer (δ very small), the neural network seems to overfit. We use the GP marginal likelihood to tune the hyperparameters of the deep neural network during training. The neural network is trained with a single hidden layer of 20 units and we apply a nonlinear `tanh` activation function. Our goal is to determine the best regularisation parameter δ to trade off between underfitting and overfitting. More precisely, Figure 5.1 shows the training log marginal likelihood of the GP-predictive model using the GGN approximation. We illustrate the *mean squared error* (MSE) along with the training and testing losses. The black star indicates the optimal hyperparameter according to the test loss and the training log marginal likelihood. The marginal likelihood chooses the hyperparameter value for which the test error is the lowest. In order to evaluate the quality of the trade-off on the predictive, we can perform different covariance structures. That is, in Figure 5.2 we illustrate the generalized linear model predictives (predictive means) under the three variants of covariance matrix structures: full, diagonal and KFAC (Kronecker-factored approximate curvature). On the upper panel we evaluate the *underfit model*; below, the *overfit model*. We depict in this figure the trade-off between complexity and capacity. The *overfit model* tries to fits the noise while the *underfit model* fails to provide clear confidence. Similarly, in the classification case with *Two Moons* dataset shown in Figure 5.3 we obtain similar results, the optimal model exhibiting better results; however, the decision boundary becomes wider farther away from the data. In the next section we continue our analysis of the approximate posterior predictive distribution.

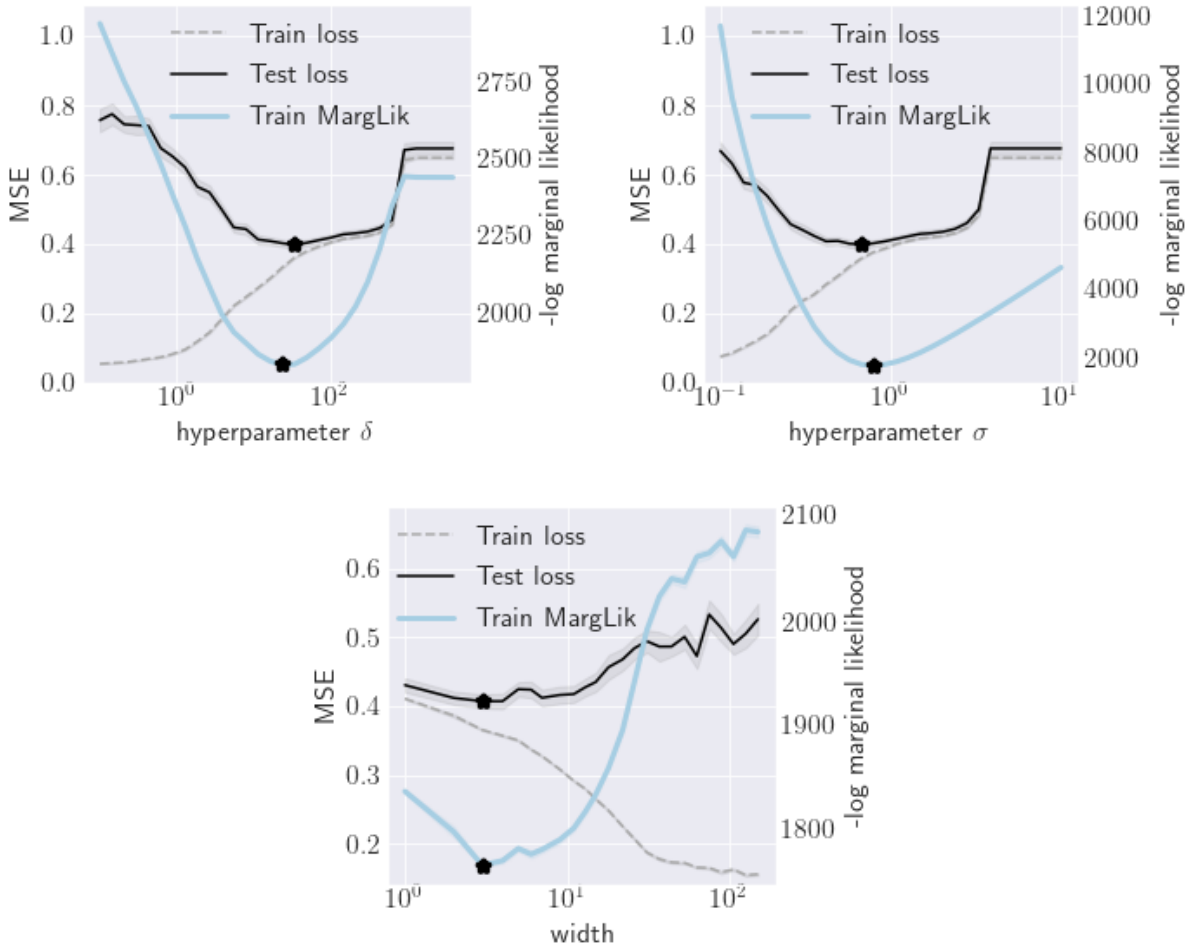

Figure 5.1 Regression: Hyperparameter tuning using the GP with Laplace approximation for model selection on *Wine* dataset.

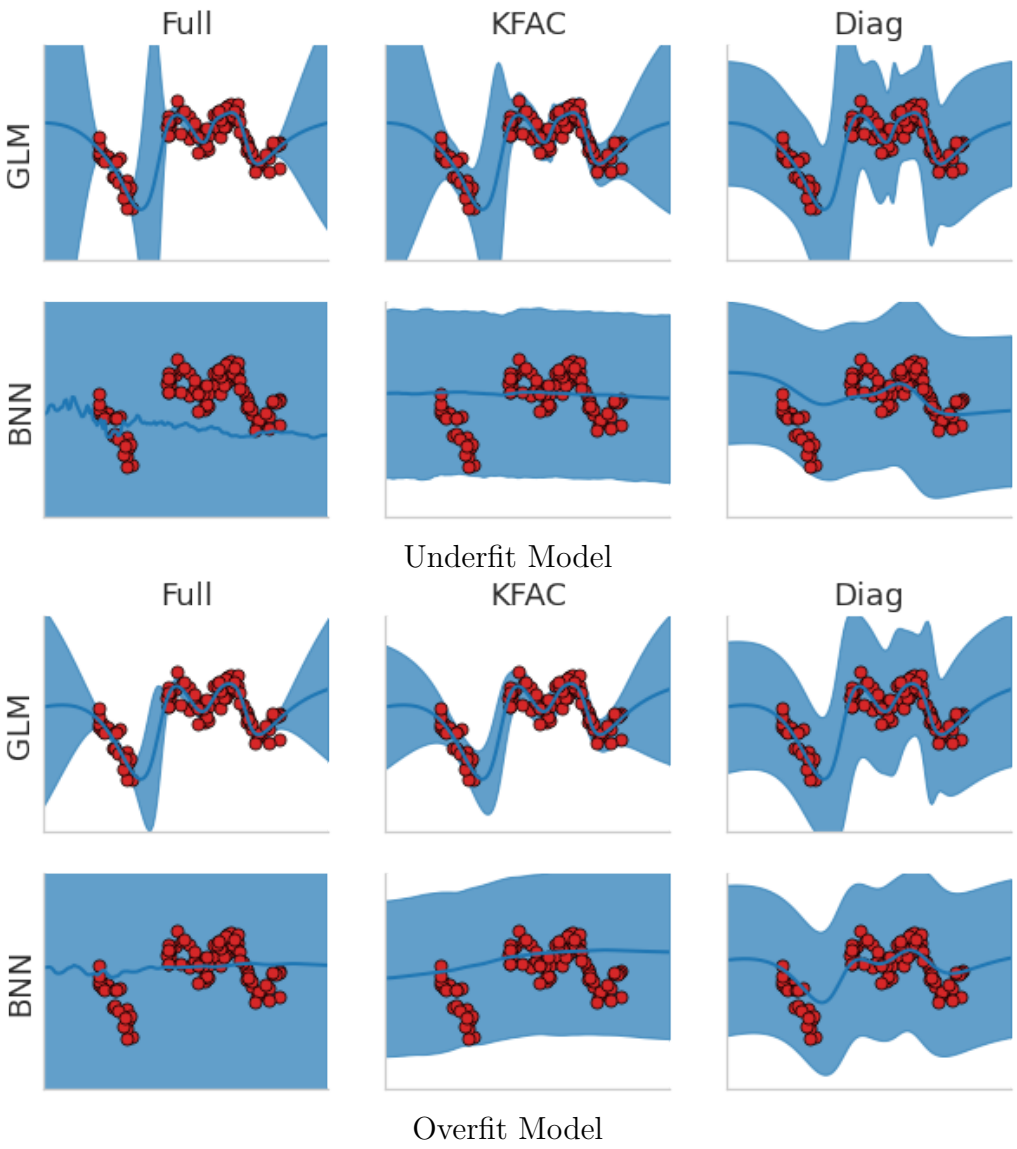

Figure 5.2 Regresion: Predictive Means for regression case: GLM-GGN solves the underfitting and overfitting problems on *Wine* dataset.

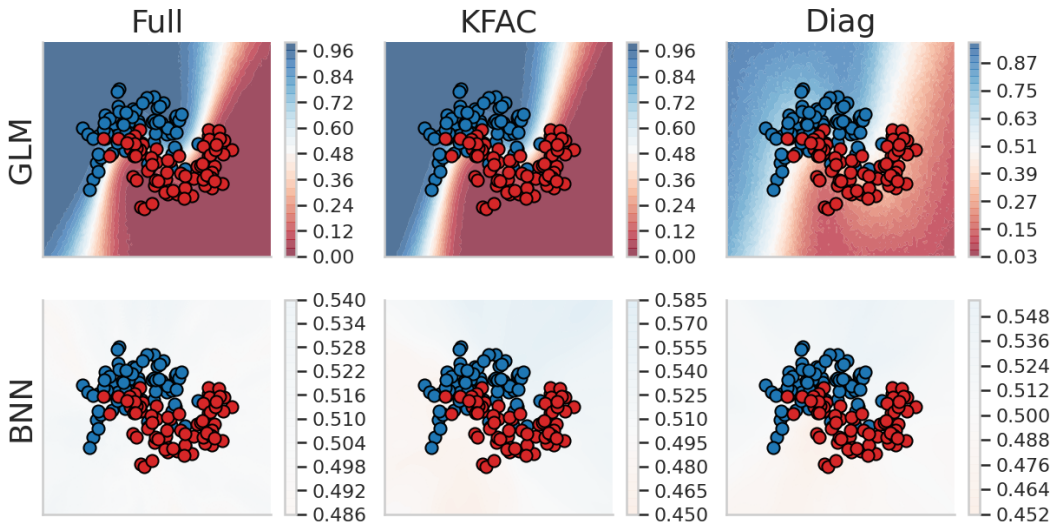

Figure 5.3 Classification: Predictive means $\mathbb{E}[\mathbf{y}_\star|\mathbf{x}_\star, \mathcal{D}]$ under the three approximation variants on *Two Moon* dataset.

5.2 Posterior Predictive Distributions

We consider toy datasets, *Two moons* and *Banana* for the classification task where we are given inputs $\mathbf{x}_n \in \mathbb{R}^2$ and targets $y_n \in \{0, 1\}$, and *Snelson* for the regression task where we are given inputs $x_n, y_n \in \mathbb{R}$. These datasets [31] are well known in the literature for training toy problems to benchmark nonlinear models such as neural networks. Both tasks use a *multilayer perceptron* with three layers and bias parameters, 25 hidden units per layer, and a `tanh` activation function. The neural network function $\mathbf{f}(\mathbf{x}; \mathbf{w})$ with parameter vector $\mathbf{w} \in \mathbb{R}^P$ maps the inputs to the outputs. For the classification case, we use a Bernoulli likelihood where the model is defined as $Y \sim \text{Bernoulli}(\mathbf{f}(\mathbf{x}; \mathbf{w}))$. For the regression case, we have an additional hyperparameter σ^2 with Gaussian likelihood defined as $Y \sim \mathcal{N}(\mathbf{f}(\mathbf{x}; \mathbf{w}), \sigma^2)$. We use a diagonal prior such that $p(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}, \mathbf{S}_0 = \delta^{-1}\mathbf{I}_P)$, where δ/N denotes the weight decay factor and is chosen based on the validation dataset structure (the negative log likelihood, architecture and method). To compute the approximation of the marginal likelihood we make use of the Laplace–GGN method. The marginal likelihood critically informs the decision on whether to choose one model over the other. We prefer a model with suitable parameters δ and σ^2; this method can replace the cross-validation technique usually employed in deep learning. [32] explored this technique in the context of Laplace approximation and variational inference with Gaussian processes. To estimate the generalization error, we use the average likelihood on the test dataset at the maximum a posteriori (MAP) estimate. We first train the network to obtain the MAP estimate with the objective

function $\frac{1}{|\mathcal{D}_{\text{test}}|}\ell(\mathbf{w}_\star, \mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n; \mathbf{w}_\star)) + \log p(\mathbf{w}_\star)$ using Adam optimizer. That is, Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moment. The choice of Adam is because it is straightforward to implement, is computationally efficient and has little memory requirements. In a second step, we compute the different posteriors and predictives with the parameters obtained after training the optimal weights $\mathbf{w}_\star$. Finally, we use our proposed methods to understand the predictions of our models. We empirically evaluate the GLM predictive for the Laplace–GGN approximated posterior and compare it to the (naive) BNN predictive. Figure 5.4 shows that the GLM predictive for the Laplace–GGN approximated posterior consistently outperforms the BNN predictive for different values of the prior precision δ.

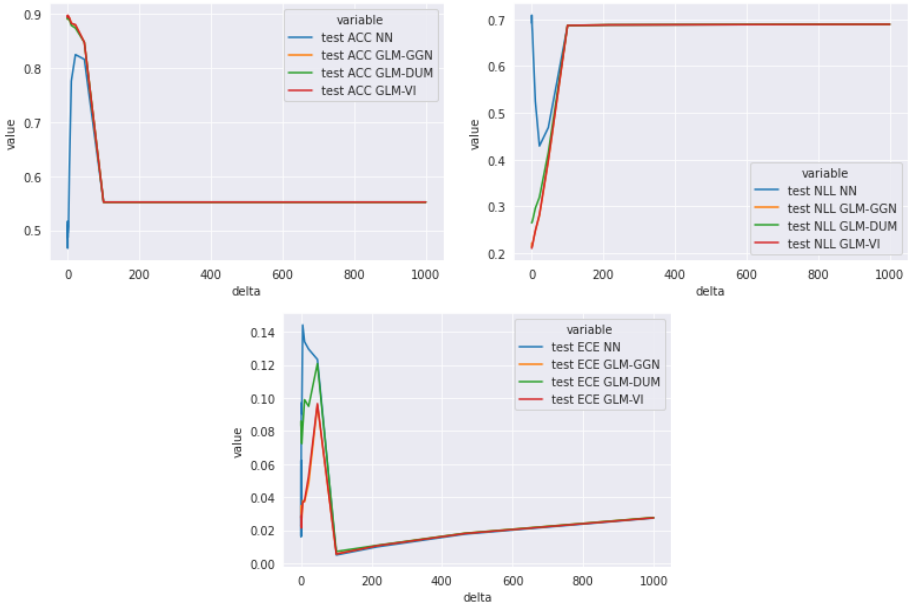

Figure 5.4 Effect of hyperparameter δ on different performance metrics for *Banana* dataset (clockwise: accuracy, negative log likelihood (NLL) and expected calibration error (ECE)).

5.3 Impact of the Generalized Gauss–Newton Method on Evidence Learning

Uncertainty quantification is often termed as *evidence learning* in deep learning; [10] provided a useful interpretation for the predictive variance, which decomposes the source of uncertainty of a model into epistemic (model noise) and aleatoric uncertainty (label noise). One way to better quantify the aleatoric uncertainty (due to the presence of noise in the data) is by increasing the sample size of the data; reducing this uncertainty is not possible. However, this procedure could reduce the epistemic uncertainty due to lack of knowledge of the data. This

trade-off will allow us to establish to what extent a model is uncertain about its predictions. This decomposition could be useful in the case of *continual learning*, where the data in the old tasks are no longer available for training new ones. [33] proposed a formal decomposition of two sources of uncertainty of the total variance, which can be written as

$$Var_{p_{\mathcal{GL}}(\mathbf{y}_\star|\mathbf{x}_\star,\mathcal{D})}(\mathbf{y}_\star) = \underbrace{\int \left(\mathbb{E}_{p_{\mathcal{GL}}(\mathbf{y}_\star|\mathbf{x}_\star,\mathbf{w})}(\mathbf{y}_\star) - \mathbb{E}_{p_{\mathcal{GL}}(\mathbf{y}_\star|\mathbf{x}_\star,\mathcal{D})}(\mathbf{y}_\star) \right)^2 q(\mathbf{w}) d\mathbf{w}}_{\text{epistemic uncertainty}} + \underbrace{\int Var_{p_{\mathcal{GL}}(\mathbf{y}_\star|\mathbf{x}_\star,\mathbf{w})}(\mathbf{y}_\star) q(\mathbf{w}) d\mathbf{w}}_{\text{aleatoric uncertainty}} .$$

In practice, to analyze the model confidence we inspect the uncertainty in the posterior distribution: in most cases, the epistemic uncertainty is low outside the data when it is distant from the decision boundary, due to the non-stationary kernel. In this section we analyze the model confidence of different predictive approximate posteriors in the Bayesian neural network case. We analyze the naive BNN and the GLM predictive under different covariance posteriors: full, Kronecker-factored (KRON or KFAC), or diagonal. We consider the classification case of the Banana datatset from UCI [31], using a neural network of two hidden layers with 50 units each and applying the `tanh` activation function. Figure 5.6 illustrates the different Bayesian neural network predictives for the classification task. We see that the naive BNN predictive suffers from underfitting compared to the GLM. That is, the model is unable to display the decision boundary and exhibits huge variance. Underfitting is due to samples from the mismatched region of the posteriors. The GLM method resolves the underfitting problem, the predictive variance increasing when distant from the data. Interestingly, we see that the GLM model decomposes the predictive variance into meaningful uncertainty quantities: the aleatoric uncertainty, which is data-inherent, shows a clear separation between classes identified by the decision boundary, while the epistemic uncertainty, which is model-specific, increases away from the data. The Kronecker-factored (KRON) posterior is capable making a clear separation between classes even under the naive BNN model. These results are in line with [33] findings. That is, the GLM predictive is adaptive to deep neural networks where the model is more involved (nonlinear) due to a more complex architecture and presents good performance results shown in Figure 5.5 with the *Banana* dataset. As claimed in the previous experiments, the naive BNN predictive underfits in comparison to the GLM. For the GLM, the underfitting is resolved due to the linear structure of the predictive. In fact, we see that the GLM predictive with Laplace–GGN posterior decomposes the variance

into an aleatoric uncertainty at class boundaries and epistemic uncertainty when less data is present. This proves that the GLM/Laplace–GGN approach adapts for deeper (more non-linear) architecture and yields better qualitative results, whereas the BNN approach yields worse results when the neural network is deeper.

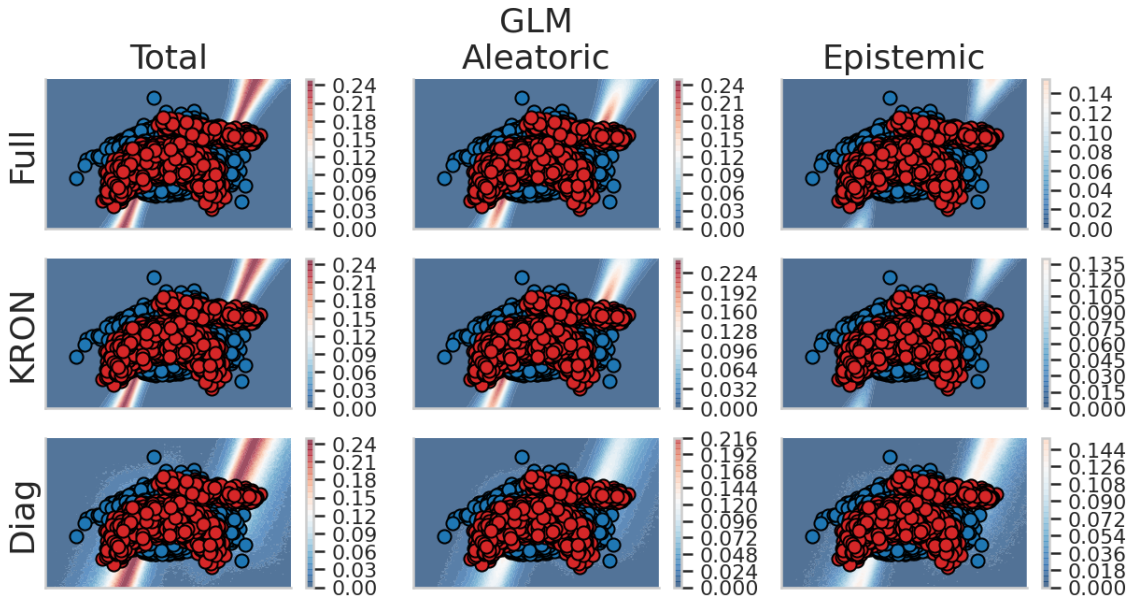

Figure 5.5 Classification: (harder task) Decomposition of uncertainty $Var[\mathbf{y}_\star|\mathbf{x}_\star, \mathcal{D}]$ for GLM-GGN model: Epistemic and aleatoric uncertainty with two hidden layers and 50 units using a `tanh` activation function on *Banana* dataset.

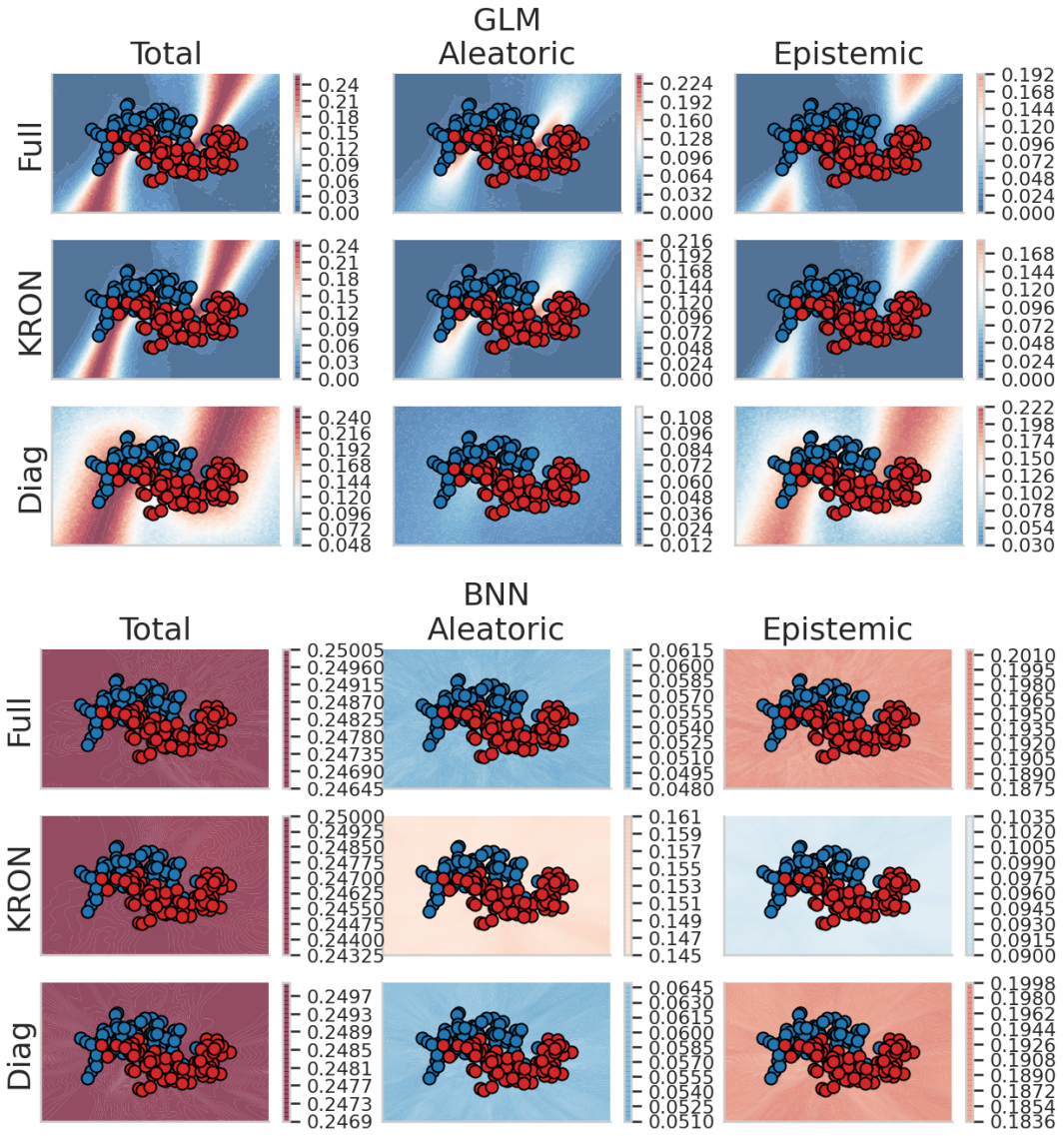

Figure 5.6 Classification: Decomposition of uncertainty $Var[\mathbf{y}_\star|\mathbf{x}_\star, \mathcal{D}]$ for GLM (top) vs BNN (bottom): epistemic and aleatoric uncertainty with two hidden layers and 50 units using a `tanh` activation function on *Two-Moons* dataset.

Figure 5.7 demonstrates the Laplace–GGN approximation under the GLM method yields to more confident predictions. In particular, for the GLM/Laplace–GGN approach with diagonal posterior, we see a less confident behaviour. In other words, the epistemic and the aleatoric uncertainty are not very meaningful in the case of the diagonal posterior. One way to explain this behaviour is due to the rigid nature of diagonal approximations, which makes them less meaningful. In fact, the diagonal approximation cannot capture all the correlations, and therefore neglects important correlations among the weights.

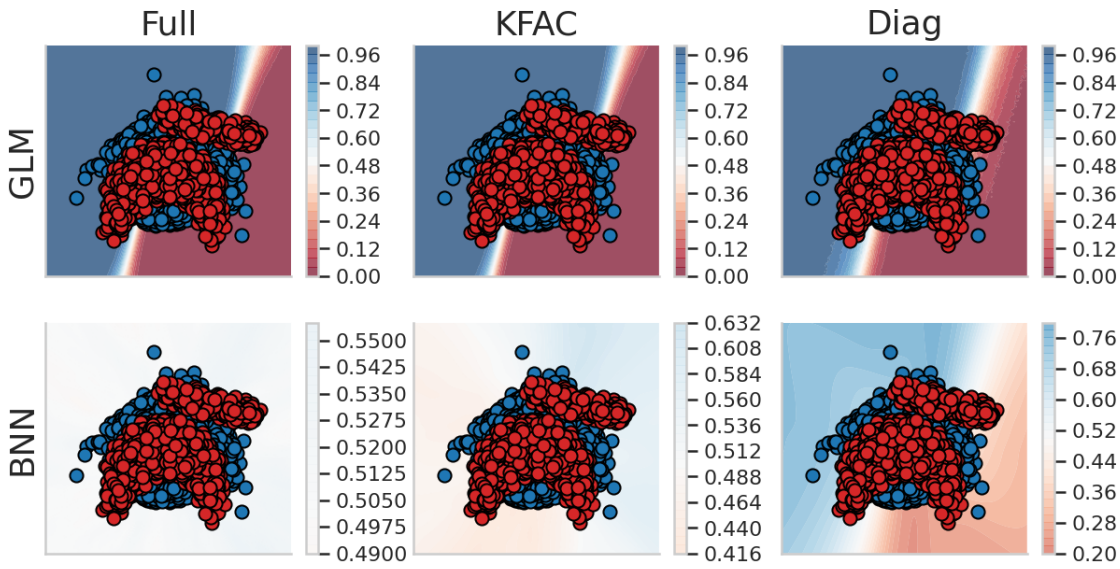

Figure 5.7 Classification: Predictive Means $\mathbb{E}[\mathbf{y}_\star|\mathbf{x}_\star, \mathcal{D}]$ under different covariance structures with two hidden layers and 50 units using a `tanh` activation function on *Banana* dataset.

5.3.1 Predictive Distribution with Variational Inference

In this section, we want to compare the quality of the posterior predictives generated by Laplace-GGN approximation and variational inference. Understanding predictions of neural networks using feature learning can help with data explainability and improves bias uncertainty by identifying inconsistency in the training data. In Figure 5.8 we consider a regression task with the *Snelson* dataset available from the UCI Machine Learning Repository. We use two hidden layers in the multilayer perceptron and 64 units with a sigmoidal transfer function. We construct the posterior predictive distributions on the regression task based on Laplace approximation and on variational inference. For the Laplace approximation, we illustrate the BNN based on GGN approximation and its corresponding Gaussian process view, denoted by BNN-GGN-GP, and we seek to compare this to the vartiational inference approximation, denoted by BNN-VOGGN. Since these two methods use different objectives, they lead to different approximate posterior predictives. We want to compare the marginal mean and variance for each model. First, we note that the BNN–Laplace approach with GGN provides reasonable uncertainty estimates with accurate predictions for both Bayesian linear regression and the GP method. In contrast, the BNN-VOGGN displays huge variance when data are missing.

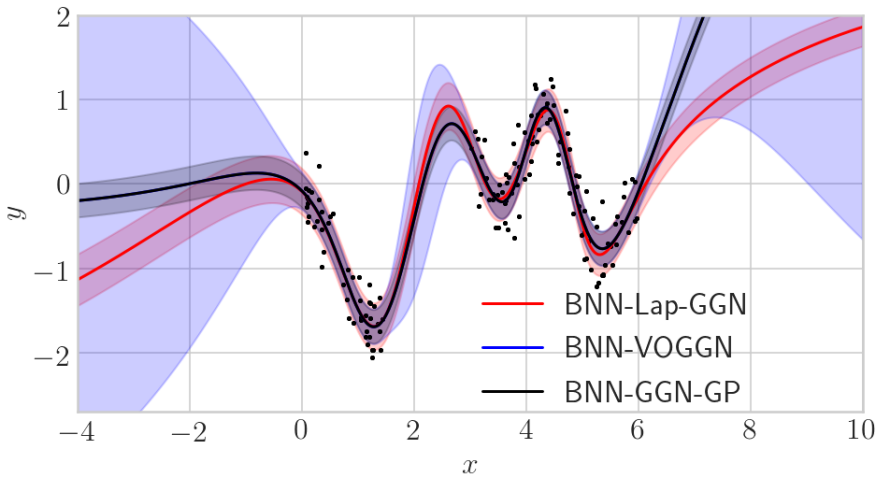

Figure 5.8 Regression: Visualization of predictive distributions based on Laplace–GGN approximation and variational inference with *Snelson* dataset.

5.4 Out-of-Distribution Detection with Generalized Gauss–Newton

5.4.1 Impact of Hyperparameter Tuning

It is worth emphasizing that different neural network architectures induce different feature maps and therefore different Jacobians. More interestingly, some prior choices lead to good inductive biases for Bayesian neural networks. Figure 5.4 shows how performance can be substantially affected by the prior choice of the hyperparameter δ. The ultimate goal is to retain the optimal model by choosing the δ that minimizes the negative log likelihood (NLL). We consider the classification *Banana* dataset available from the UCI Machine Learning Repository. We wish to compare neural network based on the naive BNN and the GLM. The GLM is constructed under the GGN-approximation with full covariance matrix, the diagonal matrix, and the GP model. We split the data into training, testing and validation datasets with proportions of 80%, 10% and 10% respectively. To avoid class imbalance, we stratified the class label and obtained a proportioned sample for each particular class. We used 16 different prior precision hyperparameters δ on a log-space grid from 0.01 to 100. The network architecture is constructed with 50 hidden units and two layers with a `tanh` activation function on each hidden layer. After selecting the optimal value for the hyperparameter δ, we report performance on test accuracy, the negative test log likelihood, and the expected calibration error (ECE). For the first metric a higher number is better while for the second and third a lower number is preferable. From Table 5.1 we notice that the GP predictive with GGN approximation outperforms the other methods on all metrics and in particular is

well calibrated according to the ECE, which measures how well the predicted uncertainty is adjusted to the empirical accuracy. In fact, the GP-GGN shows good calibration performance due its *functional inference* properties.

Table 5.1 Method Evaluations on *Banana* dataset

Method	NLL	Accuracy	ECE
BNN	0.4289	0.8251	0.1295
GLM-GGN	0.2151	0.8955	0.0288
GP-GGN	**0.2137**	**0.8968**	0.0251
GP-GGN-diag	0.2485	0.8855	**0.0226**

Table 5.2 Method Evaluations on FMNIST Image Classification

Method	FMNIST		
	Accuracy	NLL	ECE
BNN	$91.39_{\pm 0.11}$	$0.269_{\pm 0.003}$	$0.172_{\pm 0.011}$
GLM-GGN-Full	$85.60_{\pm 0.21}$	$\mathbf{0.254}_{\pm 0.032}$	$0.061_{\pm 0.034}$
GLM-GGN-KFAC	$91.88_{\pm 0.21}$	$0.274_{\pm 0.093}$	$\mathbf{0.015}_{\pm 0.071}$
GP-GGN	$\mathbf{92.18}_{\pm 0.24}$	$0.376_{\pm 0.011}$	$0.016_{\pm 0.021}$

Table 5.3 Method Evaluations on CIFAR Image Classification

Method	CIFAR		
	Accuracy	NLL	ECE
BNN	$67.19_{\pm 0.512}$	$0.567_{\pm 0.043}$	$0.255_{\pm 0.071}$
GLM-GGN-Full	$\mathbf{82.76}_{\pm 0.76}$	$\mathbf{0.342}_{\pm 0.053}$	$0.088_{\pm 0.051}$
GLM-GGN-KFAC	$80.26_{\pm 0.18}$	$0.701_{\pm 0.091}$	$0.089_{\pm 0.035}$
GP-GGN	$81.78_{\pm 0.77}$	$0.541_{\pm 0.009}$	$\mathbf{0.032}_{\pm 0.017}$

5.4.2 Scalability

For larger datasets, we use MNIST [34], FMNIST [35] and CIFAR [36] datasets for image classification. This will allow us to evaluate scalability performance for the proposed approaches. We use a naive BNN with Laplace approximation, the GLM–Laplace with full GGN approximation, the GLM with KFAC–Laplace–GGN approximation, and a GLM-GP model with a sparse posterior approximation on a subset of $M = 400$ data points. We consider a convolution neural network (CNN) architecture for the MNIST and CIFAR datasets,

training with four hidden layers with sizes of $1024, 512, 256$ and 128. The GLM predictive shows superior performance results in terms of accuracy and NLL in comparison with other methods. Interestingly, despite the fact that the GP model uses a subset of the training data to make predictions, it outperforms in terms of expected calibration error (ECE) [37] (which simply takes a weighted average over the absolute accuracy/confidence difference). This is because the GP model makes use of the full Laplace–GGN covariance matrix, whereas in the case of the parametric model, only a posterior KFAC covariance approximation is used. We compare the probabilities using the following metrics: accuracy, NLL and ECE. To evaluate the efficacy of the approach in detecting the out-of-distribution data, we evaluate the predictives on out-of-distribution (MNIST) and in-distribution (FMNIST) datasets respectively. First, we compute the predictive entropies of the output and compare the out-of-distribution and in-distribution datasets against each other. More specifically, we compute the predictive entropy of the underlying output probability (obtained by sampling). When the output distribution is a uniform distribution then the predictive entropy is at its maximum. For out-of-distribution we desire higher predictive entropies, in contrast for in-distribution data we desire lower predictive entropies. Figure 5.9 illustrates the predictive entropy density under different posterior methods using the MNIST and FMNIST datasets for out-of-distribution and in-distribution data respectively. We note that the GLM predictive outperforms the other methods. The naive BNN is overconfident while the GLM predictive is underconfident with a better out-of-distribution detection reflected by good results on calibration metrics (ECE), as shown in Tables 5.2 and 5.3. In summary, the GLM predictive outperforms based on test accuracy while the naive BNN shows the worst results. We obtain similar results with the GP-GGN model shown in Figure 5.10 for different sparse GP models as we increase the number of inducing points. In conclusion, the three tables 5.1, 5.2 and 5.3 show clearly that the Laplace approximation with GGN outperforms the naive BNN in all metrics. In a second analysis, when taking only the GLM–GGN method itself, we note that, depending on which covariance structure is used, different results are obtained. That is, the KFAC structured covariance provides a good trade-off between expressiveness and speed. Diagonal approximations perform significantly worse than KFAC but are very cheap to compute and are therefore not suggested in most cases.

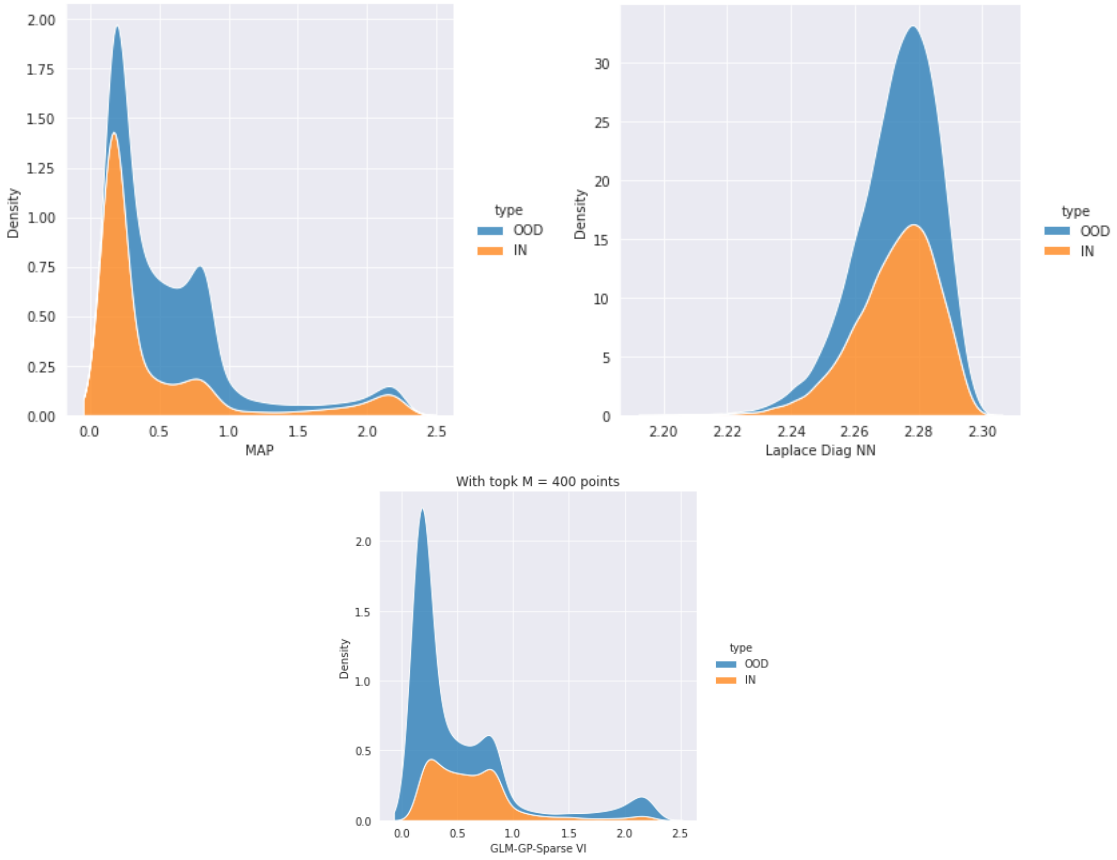

Figure 5.9 Out-of-distribution (MNIST) vs in-distribution (FMNIST) detection with CNN architecture incorporating two fully connected classification layers.

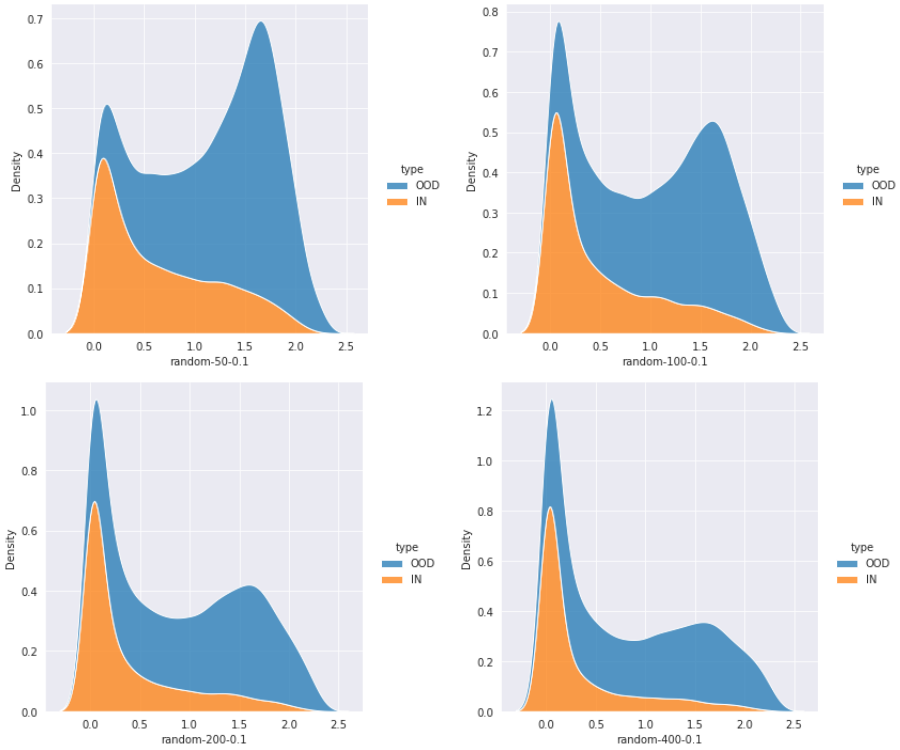

Figure 5.10 Effect on entropy of inducing points under the GP-GGN model: out-of-distribution (MNIST) vs in-distribution (FMNIST) with CNN architecture incorporating two fully connected classification layers.

CHAPTER 6 FUTURE WORK

In this thesis we have complemented our theoretical results with experiments that can be used for future work. In fact, the generalized Gauss–Newton method and approximate inference in Bayesian deep learning can be combined to extend theoretical work and to investigate convergence, the choice of priors and domain representation.

We have presented different approximations that rely on the Gauss–Newton approximation, each having its own strengths and weaknesses. Approaching Bayesian deep learning with these approximate inferences from a probabilistic perspective allows us to understand how accurate these approximations are. Traditionally, Bayesian deep learning methods rely heavily on MAP estimation which plays a role as a weight space regularizer, but it is worth investigating how we could add perturbations to this regularizer in function space. This formulation has already been discovered but not as yet extended to a large scale. One particular example is that the approximation of this regularizer in function space yields better performance results than classical weight methods discussed in the application of *continual learning* by [23]. We have shown that GLMs become GPs due to the Gauss–Newton method, which permits analysis of the form of the Jacobians under different neural network architectures and helps us to understand GP kernel structures. A more recent but still ambiguous development is the understanding of the structure of posterior covariance approximations, especially for low-rank covariances. A key advantage of linear transformation from a probabilistic neural network to a GLM or a GP model is the reduction of the computational complexity. This enables us to obtain a closed form of the functional posterior for regularizing neural networks in the function space as opposed to the weight or parameter space. In addition, the function-space approximation leads to the same posterior predictive as a full posterior covariance in the parametric space and therefore enables an interesting application in transfer learning.

CHAPTER 7 CONCLUSION AND RECOMMENDATIONS

In this thesis, we have given an account of Bayesian deep learning under approximate inference using the generalized Gauss–Newton method. In particular, approximating the posterior of the GLM gives us an approximate formulation of the neural network. Using this formulation the GLM becomes a Gaussian process, allowing identification of a function-space posterior approximation for neural networks. Interestingly, linearizing the neural network results in a closed-form posterior predictive distribution and thereby reduces the computational cost. This work has enabled a detailed investigation into how the accuracy of approximate Bayesian inference can solve common problems for large datasets. That is, this method solves the underfitting problem from which Bayesian neural networks tend to suffer. In addition, we have shown how uncertainty quantification and out-of-distribution detection can be handled even with large datasets.

We have shown that the combination of Laplace approximation and the generalized Gauss–Newton approach yields an improvement in BNN learning. The process is accomplished in two principal steps. The first step consists in optimizing the neural network with the MAP (maximum a posteriori) estimate; the second step applies GGN, which linearlizes the neural network at the MAP estimate. This approximation allows us to identify the GLM via moment matching between the Laplace and the GLM likelihood when the likelihood is Gaussian. In particular, this GLM turns into a Bayesian linear regression model and we therefore obtain a posterior approximation of the neural network. Interestingly, this GLM can easily identify a Gaussian process model, enabling us to obtain a closed-form posterior predictive and marginal likelihood. This connection supports inference in the function space under Laplace-GGN posterior approximation, a formulation that is interesting when the number of parameters is much larger than the number of data points. In fact, we have two views, the GLM model in the weight space and the GLM-GP in the function space. We have shown for Bayesian linear regression that these models are both useful for the posterior predictive and marginal likelihood computation in the Bayesian model selection case, and have provided robust predictions even though we do not have the true likelihood. That is, we illustrated in the regression and classification cases that using the GLM yields more consistent posterior distributions than neural network sampling. In addition, the marginal likelihood approximation to the GLM creates an automated way to select the optimal hyperparameters of the neural network in the training phase.

Variational inference remains a direct competitor to the Laplace approximation for deep

learning algorithms, differing from the latter on how to update the variational parameters; basically we need to compute an expectation to obtain the updates and identify the stationarity conditions generally approximated by sampling. We showed by applying the GGN in the VI framework we can derive a new algorithm: the variational online generalized Gauss–Newton algorithm. At each step, VOGGN samples the neural network and obtains different neural network feature maps, leading to a better generalization of the predictive approximation. Despite its accurate predictions based on the posterior approximation, VOGGN shows huge variance when data are missing. In summary, both variational inference and Laplace–GGN approximation algorithm provide consistent results for uncertainty quantification and out-of-distribution detection. However, a significant issue from which these algorithms suffer is their highly correlated parameters. In fact, they fail to capture correlations with a Gaussian approximating family and are difficult to scale to large models due to computational costs and high variance of gradient updates. [38] proposed adding correlation in the hidden units. This technique could potentially reduce the gap between the true and the corresponding variational posterior, which might improve performance.

REFERENCES

[1] G. E. Hinton and D. v. Camp, "Keeping neural networks simple by minimizing the description length of the weights," 1293. [Online]. Available: http://www.cs.toronto.edu/~fritz/absps/colt93.pdf

[2] R. M. Neal, "Bayesian learning for neural networks," Ph.D. dissertation, 1995.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[4] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[5] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. 65, pp. 1939–1959, 2005. [Online]. Available: http://jmlr.org/papers/v6/quinonero-candela05a.html

[6] M. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," *AISTATS*, 2009.

[7] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 4, pp. 1303–1347, 2013. [Online]. Available: http://jmlr.org/papers/v14/hoffman13a.html

[8] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*. MIT Press, 2021. [Online]. Available: probml.ai

[9] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[11] M. E. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava, "Fast and scalable Bayesian deep learning by weight-perturbation in adam," 2018.

[12] R. Price, "A useful theorem for nonlinear devices having Gaussian inputs," *IRE Transactions on Information Theory*, vol. 4, no. 2, pp. 69–72, 1958.

[13] G. Bonnet, "Transformations des signaux aléatoires à travers les systèmes non linéaires sans mémoire," *Annales des Télécommunications*, vol. 19, pp. 203–220, 1964.

[14] M. Opper and C. Archambeau, "The variational Gaussian approximation revisited," *Neural Comput.*, vol. 21, no. 3, pp. 786–792, 2009. [Online]. Available: https://doi.org/10.1162/neco.2008.08-07-592

[15] H. Ritter, A. Botev, and D. Barber, "A scalable Laplace approximation for neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Skdvd2xAZ

[16] M. E. Khan and H. Rue, "The Bayesian learning rule," 2021. [Online]. Available: https://www.arxiv-vanity.com/papers/2107.04562/

[17] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse, "Noisy natural gradient as variational inference," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5852–5861. [Online]. Available: http://proceedings.mlr.press/v80/zhang18l.html

[18] L. Bottou, F. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[19] J. Martens and R. Grosse, "Optimizing neural networks with Kronecker-factored approximate curvature," 2020. [Online]. Available: https://dl.acm.org/doi/10.5555/3045118.3045374

[20] P. McCullagh and J. Nelder, *Generalized Linear Models, Second Edition*, ser. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall, 1989. [Online]. Available: http://books.google.com/books?id=h9kFH2_FfBkC

[21] A. Chan and D. Dong, "Generalized Gaussian process models," 06 2011, pp. 2681–2688.

[22] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as Gaussian processes," 2018. [Online]. Available: https://arxiv.org/abs/1711.00165

[23] P. Pan, S. Swaroop, A. Immer, R. Eschenhagen, R. E. Turner, and M. E. Khan, "Continual deep learning by functional regularisation of memorable past," H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 4453–4464. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/2f3bbb9730639e9ea48f309d9a79ff01-Paper.pdf

[24] M. E. Khan, A. Immer, E. Abedi, and M. Korzepa, "Approximate inference turns deep networks into Gaussian processes," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/b3bbccd6c008e727785cb81b1aa08ac5-Paper.pdf

[25] D. J. C. MacKay, "The Evidence Framework Applied to Classification Networks," *Neural Computation*, vol. 4, no. 5, pp. 720–736, 09 1992. [Online]. Available: https://doi.org/10.1162/neco.1992.4.5.720

[26] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," Lille, France, pp. 1613–1622, 07–09 Jul 2015. [Online]. Available: https://proceedings.mlr.press/v37/blundell15.html

[27] H. Ritter, A. Botev, and D. Barber, "A Scalable Laplace approximation for neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Skdvd2xAZ

[28] K. Osawa, S. Swaroop, M. E. E. Khan, A. Jain, R. Eschenhagen, R. E. Turner, and R. Yokota, "Practical deep learning with Bayesian principles," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/b53477c2821c1bf0da5d40e57b870d35-Paper.pdf

[29] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[30] A. Mishkin, F. Kunstner, D. Nielsen, M. Schmidt, and M. E. Khan, "Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/d3157f2f0212a80a5d042c127522a2d5-Paper.pdf

[31] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[32] F. D. Foresee and M. Hagan, "Gauss-Newton approximation to Bayesian learning," *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 3, pp. 1930–1935 vol.3, 1997.

[33] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics Data Analysis*, vol. 142, p. 106816, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016794731930163X

[34] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," *http://yann.lecun.com/exdb/mnist/*, 2010. [Online]. Available: https://ci.nii.ac.jp/naid/10027939599/en/

[35] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017, cite arxiv:1708.07747Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/. [Online]. Available: http://arxiv.org/abs/1708.07747

[36] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[37] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using Bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, p. 2901–2907.

[38] M. Tomczak, S. Swaroop, and R. Turner, "Efficient low rank Gaussian variational inference for neural networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 4610–4622. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/310cc7ca5a76a446f85c1a0d641ba96d-Paper.pdf

[39] C. Williams and D. Barber, "Bayesian classification with Gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1342 –1351, Dec. 1998.

[40] T. P. Minka, "Expectation propagation for approximate Bayesian inference," *CoRR*, vol. abs/1301.2294, 2013. [Online]. Available: http://arxiv.org/abs/1301.2294

[41] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, p. 859–877, Apr 2017. [Online]. Available: http://dx.doi.org/10.1080/01621459.2017.1285773

[42] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

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

APPENDIX A GAUSSIAN PROCESSES

We define the general case for a regression task where we assume the observations follow a Gaussian distribution

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{f}, \sigma_n^2\right).$$

The main goal is to compute the posterior distribution of the latent Gaussian process (GP); applying Bayes' rule we have

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{p(\mathbf{y})}. \tag{A.0.1}$$

The posterior distribution of the latent function given the data is a Gaussian,

$$\mathbf{f}|\mathcal{D}, \boldsymbol{\lambda} \sim \mathcal{N}\left(\mathbf{K_{ff}}\left(\mathbf{K_{ff}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}, \mathbf{K_{ff}} - \mathbf{K_{ff}}\left(\mathbf{K_{ff}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{K_{ff}}\right),$$

where $\boldsymbol{\lambda}$ denotes the kernel hyperparameters and $\mathbf{K_{ff}}$ is the covariance matrix induced by the covariance function $\boldsymbol{\kappa}(.,.;\boldsymbol{\lambda})$ evaluated at every pair of inputs. When the likelihood and the posterior of the latent function are both Gaussian distributed, integration over a product of Gaussians produces a Gaussian distribution. For test points $\mathbf{x}_\star$, we make a prediction for new labels $\mathbf{y}_\star$ as follows:

$$\mathbf{y}_\star|\mathbf{x}_\star, \mathcal{D}, \boldsymbol{\lambda}, \sigma_n^2 \sim \mathcal{N}\left(\boldsymbol{\mu}(\mathbf{x}_\star), \boldsymbol{\Sigma}(\mathbf{x}_\star, \mathbf{x}'_\star) + \sigma_n^2\mathbf{I}\right),$$

where

$$\boldsymbol{\mu}(\mathbf{x}_\star) = \boldsymbol{\kappa}(\mathbf{x}_\star, \mathbf{X})\left(\mathbf{K_{ff}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}$$

and

$$\boldsymbol{\Sigma}(\mathbf{x}_\star, \mathbf{x}'_\star) = \boldsymbol{\kappa}(\mathbf{x}_\star, \mathbf{x}_\star) - \boldsymbol{\kappa}(\mathbf{X}, \mathbf{x}_\star)\left(\mathbf{K_{ff}} + \sigma_n^2\mathbf{I}\right)^{-1}\boldsymbol{\kappa}(\mathbf{X}, \mathbf{x}_\star).$$

We note that the mean and the kernel depend on the model's hyperparameter vector $\boldsymbol{\lambda}$, which governs the quality of the Gaussian process's fit to the data. Estimating these parameters

requires the marginal likelihood of the data,

$$p(\mathcal{D}|\boldsymbol{\lambda}, \sigma) = \int p(\mathbf{y}^\star|\mathbf{f}, \sigma_n^2) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\lambda}) d\mathbf{f}; \tag{A.0.2}$$

integrating (marginalizing) over the latent function values $\mathbf{f}$ gives the marginal distribution

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\lambda}, \sigma^2) = \mathcal{N}\left(\mathbf{0}, \mathbf{K_{ff}} + \sigma_n^2 \mathbf{I}\right).$$

For convenience of optimization we work with the log marginal likelihood

$$\log p(\mathcal{D}|\boldsymbol{\lambda}, \sigma^2) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K_{ff}} + \sigma_n^2| - \frac{1}{2}\mathbf{y}^T\left(\mathbf{K_{ff}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}.$$

In the presence of a closed-form solution, the log marginal likelihood is computed through efficient gradient-based optimization techniques such as L-BFGS or conjugate gradients. When the likelihood $p(\mathbf{y}|\mathbf{f}, \boldsymbol{\lambda})$ is non-Gaussian (due to the loss of conjugacy) the posterior distribution of the latent function given the data $p(\mathbf{f}|\mathcal{D}, \boldsymbol{\lambda})$ does not have a closed-form solution; approximation-based inference could be applied to approximate $p(\mathbf{f}|\mathbf{X}, \boldsymbol{\lambda})$ in equation (A.0.2) but we need to deal with two fundamental components; first, the cost of computing the prior and the posterior, and second, the likelihoods that are not conjugate to the Gaussian distribution. Despite having an analytic closed form, using these equations quickly becomes intractable even for small datasets. The issue comes from the $\mathcal{O}(N^3)$ computational cost for computing the inverse and determinant of $\mathbf{K_{ff}}$. Usually, Bayesian models have tractable priors and likelihoods, and any intractability comes from the fact that there is no closed-form solution for the marginal likelihood, referred to as the normalizing constant of the posterior. The prior $p(\mathbf{f})$ could also be intractable, as this density already contains the inverse and the determinant calculations. Popular approximation techniques have been proposed such as Laplace approximation [39], expectation propagation [40] and variational inference [9]. Alternatively, instead of seeking a tractable approximation (of the posterior), we could sample from it and use the sample as a stochastic approximation, evaluating the marginal distribution through Monte Carlo integration. The most popular sampling methods are Markov chain Monte Carlo (MCMC) and Gibbs sampling. However, these methods scale poorly even for moderately large datasets, especially when the posterior distribution is high-dimensional. We can approach the problem by variational inference, however, solving it using sparse approximation.

APPENDIX B VARIATIONAL INFERENCE WITH GAUSSIAN PROCESS MODELS

We aim to achieve two goals: efficient computation and tractability. To deal with the intractable posterior in (A.0.1) we employ the variational inference formalism developed by [41]. The tractability issue is solved via an optimization problem based on finding the closest approximate distribution according to the Kullback–Leibler divergence measure. In information theory, to approximate an unknown distribution p from an arbitrary distribution q, a frequently used dissimilarity measure between p and q is the forward KL divergence or relative entropy [42], defined as:

$$\begin{aligned} D_{KL}(p\|q) &= \int p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} \\ &= \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} \\ &= -\mathcal{L}_e(p) + \mathcal{L}_c(p, q), \end{aligned}$$

where $\mathcal{L}_e(p)$ is the *entropy* of the distribution p and $\mathcal{L}_c(p, q)$ is the *cross-entropy* between p and q. Similarly, the *reverse KL divergence* $D_{KL}(q\|p)$ can be defined as

$$\begin{aligned} D_{KL}(q\|p) &= \int q(\mathbf{x}) \ln \frac{q(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x} \\ &= \int q(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \int q(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \\ &= -\mathcal{L}_e(q) + \mathcal{L}_c(q, p). \end{aligned}$$

To see why the D_{KL} divergence can be used as a dissimilarity measure, one can use *Jensen's inequality*, see [10]. (The D_{KL} divergence is not a distance measure since $D_{KL}(p\|q) \neq D_{KL}(q\|p)$; hence it is not symmetric [43].) The key idea of the variational inference formalism proposed by [41] is to choose a family of tractable variational distributions $q(\mathbf{f}; \boldsymbol{\lambda})$ (where $\boldsymbol{\lambda}$ corresponds to the variational parameters) to approximate the true posterior $p(\mathbf{f}|\mathbf{y})$. We determine the closest family member by minimizing the Kullback–Leibler divergence with respect to $\boldsymbol{\lambda}$:

$$D_{KL}(q(\mathbf{f}; \boldsymbol{\lambda})||p(\mathbf{f}|\mathbf{y})) = \mathbb{E}_{q(\mathbf{f};\boldsymbol{\lambda})} \left[\log q(\mathbf{f}; \boldsymbol{\lambda}) - \log p(\mathbf{f}|\mathbf{y})\right].$$

In most cases we cannot compute the KL divergence, so we need an alternative objective to optimize. A clever way to do this is by minimizing the KL divergence, which is equivalent to maximizing the lower bound of the log marginal likelihood (ELBO) $\mathcal{L}$ given by

$$\log p(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{f};\boldsymbol{\lambda})}\left[\log p(\mathbf{y},\mathbf{f}) - \log q(\mathbf{f};\boldsymbol{\lambda})\right] := \mathcal{L}(\boldsymbol{\lambda}).$$

Therefore we have two challenges, model complexity and tractability. The latter deals with an intractable posterior and can be resolved using variational inference, while the former deals with the issue of matrix inversion and can be resolved using covariance approximation.