Advances in Deep Gaussian Processes : calibration and sparsification

Gia-Lac Tran

ADVANCES IN DEEP GAUSSIAN PROCESSES: CALIBRATION AND SPARSIFICATION.

Gia-Lac Tran

A doctoral dissertation submitted to:

SORBONE UNIVERSITY

In Partial Fulfillment of the Requirements for the Degree of:

Doctor of Philosophy

Specialty : Computer Science

Supervised by: **Prof. Maurizio Filippone**

Jury:

Reviewers:

Prof. Marco Lorenzi - Inria, Sophia Antipolis - France

Prof. Annalisa Barla - University of Genève, Genova - Italie

Examiners:

Prof. Pietro Michiardi - Eurecom, Biot - France

Prof. Serena Villata - University of Côte d'Azur, Biot - France

Contents

1 Introduction **1**
1.1 Overview . . . 1
1.2 Extensions and Open Problems . . . 4
1.3 Outline and Contributions of Thesis . . . 6

2 Gaussian Processes for Big Data **8**
2.1 Overview . . . 8
2.2 Gaussian Processes . . . 9
2.2.1 Gaussian Processes for Regression . . . 9
2.2.2 Covariance function . . . 11
2.2.3 Non-Gaussian Likelihoods . . . 14
2.2.4 Limitations of Gaussian Processes . . . 14
2.3 Inducing Point Approximations . . . 16
2.3.1 Prior approximation . . . 16
2.3.2 Posterior Approximations . . . 22
2.3.3 Structure Exploiting Approximations . . . 27
2.4 Random Feature Approximations . . . 33
2.4.1 Spectral Representations . . . 34
2.4.2 Random featured-based Gaussian Processes. . . . 36
2.5 Local Approximation . . . 38

3 Calibrating Deep Convolutional Gaussian Processes **41**
3.1 Introduction . . . 41
3.2 Related Work . . . 43
3.3 On calibration of Convolutional GPs . . . 45
3.4 Proposed Method . . . 46
3.4.1 Random Feature Expansions . . . 47
3.4.2 End-to-end learning . . . 48
3.4.3 Extensions . . . 50
3.5 Experiments . . . 52
3.5.1 Reliability diagrams . . . 53
3.5.2 Extension with Deep GPs . . . 54
3.5.3 Knowing when the model does not know . . . 54
3.5.4 Extension with the SORF . . . 55
3.6 Mathematical details and other experiments . . . 56
3.6.1 Random Feature Expansion of the RBF Covariance . . . 56
3.6.2 Variational Inference for the Proposed Model . . . 57

3.6.3 Variational inference of filters in GPDNN 60
3.6.4 Reliability diagrams 60
3.7 Conclusions 62

4 Local and Global Approximation of Gaussian Processes 66
4.1 Introduction 66
4.2 Related work and background 68
4.2.1 Scalable Variational Gaussian Processes 69
4.3 Sparse within Sparse Gaussian Processes 70
4.3.1 Lower bound on marginal likelihood 71
4.3.2 H-nearest inducing inputs 72
4.3.3 Complexity 74
4.4 Experiments 75
4.4.1 Increasing the number of neighbors 76
4.4.2 Increasing the number of inducing points 76
4.4.3 Running time 78
4.4.4 Large-scale problems with a huge number of IPs 80
4.4.5 Comparison to Local GPs 81
4.5 Other results 82
4.5.1 Various options for H-nearest inducing points selection 82
4.5.2 Further visualizations on 1D examples 83
4.6 Conclusions 84

5 Conclusion 85
5.1 Themes and Contributions 85
5.2 Future work 86
5.2.1 Calibrated GP regression 87
5.2.2 Elegant mixtures of CNNs and GPs 87
5.2.3 Adaptability to online machine learning 88

Bibliography 89

1 Introduction

1.1 Overview

Machine Learning as an inductive problem. Machine Learning (ML) is seen as an application of Artificial Intelligence. The use of learning algorithms equips systems with the ability to automatically acquire helpful information from experience without being explicitly programmed. Depending on the contextual scenarios, ML has been categorized into different approaches, e.g. supervised, unsupervised, semi-supervised or by reinforcement. In this dissertation, we mainly focus on supervised learning problems where pairs of input and outputs are collected to learn a mapping functions from an input space to an output space. From the available observations, we wish to derive a function that models the underlying mapping from the input data (covariates) to labels (or target values); from the function, we can then make predictions for all possible input values. It is obvious that the problem at hand is inductive. The approaches for learning the mappting function in a given task can be grouped into two categories: parametric and non-parametric.

Parametric Modeling. Traditionally, we can use parametric machine learning algorithms to deal with supervised problems. This kind of modeling restricts the underlying mapping to a family of functional forms which is parameterized by a finite set of parameters. It also implies that no matter how much data is fed to a parametric model, it will not change its mind about how many parameters it needs (Russell and Norvig, 2003). Such parametric models often perform inefficiently if the functional form is inadequate to represent the actual unknown underlying correlation between inputs and its labels. One may be tempted to employ a flexible functional form, e.g. we can assume the parametric function is the one obtained by a neural network, but this runs into the danger of overfitting, so that we can obtain a good fit to training data, but perform badly in predictions.

Non-parametric Modeling. In contrast to parametric model accompanying with a specified functional form, algorithms using free-form mapping functions are classified as non-parametric machine learning algorithms, such as k-Nearest Neighbor (Cover and Hart, 2006), Decision Trees (Quilan, 1988) or Support Vector Machine (Cortes and Vapnik, 1995) or Kernel methods (Hofmann et al., 2008). Non-parametric feature extraction algorithms have more advantages than parametric ones and are well suited for non-normally distributed data along with being able to extract more features than the classic linear discriminant analysis (Russell and Norvig, 2003; Yang et al., 2010). In general, such non-parametric models possessing an infinite set of parameters are capable of fitting any complicated functional form. Nevertheless, it also implies that the number of labeled data required by non-parametric approaches to estimate the mapping function is greater than the parametric model with a finite set of parameters. Therefore, non-parametric models are easy prone to overfitting, especially when labeled data is scarce.

Scalability of Non-parametric approaches. In the era of big data, non-parametric models are promising solutions allowing to learn complicated patterns from data. Nevertheless, the computational complexity of non-parametric approaches depends on the training size. For example, the training phase of Kernel Support Vector Machine (Cortes and Vapnik, 1995) involves solving a quadratic problem which generally suffers cubic time complexity with respect to data size. Consider K-Nearest Neighbor (Cover and Hart, 2006) as another example; it is a non-parametric lazy learning algorithm which does not require an explicit training phase. However, K-Nearest Neighbor makes prediction on unseen data as a vote by using all the training data. Generally, the testing phase of these methods requires linear time complexity to data size. Hence, the application of non-parametric models to large-scale problems is hindered by their poor scalability.

Needs of Predictive Uncertainty. The problem of enhancing the safety of decision-making system by acting on the model's prediction in an informed manner has obtained a significant attention from the machine learning community (Guo et al., 2017). Predictive uncertainty quantification has a crucial role to strengthen the safety of an AI system (Amodei et al., 2016) by acting on the model's prediction in an informed manner. This is essential to applications where the consequence of an error is serious, such as in autonomous vehicle control and medical, financial and legal fields. Hence, accurate fitting capabilities are no longer the most important aspects for evaluating the model's effectiveness.

Source of Predictive Uncertainty. Predictive uncertainty is a conflation of several separate factors: model uncertainty, data uncertainty and distributional uncertainty. Model uncertainty or epistemic uncertainty represents the uncertainty in the estimate of model's parameter given the training data. This uncertainty can be explained away given enough data. Data uncertainty or aleatoric uncertainty comes from the complication in the observations, such as class overlap, label noise, input-dependent noise. As this kind of uncertainty accompanying the nature of data, it is irreducible even if more data are collected. Distributional uncertainty appears due to the mismatch between the training and testing distribution.

Evaluation of predictive uncertainty. Evaluating the quality of predictive uncertainties is challenging as the ground-truth uncertainty estimates are unknown. Being motivated by practical applications, there are two aspects that are able to examine the plausibility of predictive uncertainty. The first notion of quality of predictive uncertainty concerns calibration (Dawid, 1982; DeGroot and Fienberg, 1983), which measures the discrepancy between subjective forecast and (empirical) long-run frequencies. Traditionally, the quality of calibration can be numerically assessed by proper scoring rules (Gneiting and Raftery, 2007), such as the Brier score (Brier, 1950). Secondly, the quality of predictive uncertainty is also obtainable using out-of-distribution examples (Hendrycks and Gimpel, 2016). For example, if a model is trained on one dataset, but is tested on a completely different dataset, the predictive uncertainty returned by the model should be high, as testing points would be distant from training points. Recently, the works of approximation of predictive uncertainty based upon ensemble learning are robust to calibration as well as the scenarios of data shift (Lakshminarayanan et al., 2017). Alternatively, a plethora of works revolves around the Bayesian formalism (Bernardo and Smith, 2000) with the aim of adapting neural networks to encompass predictive uncertainty and give them a probabilistic flavor (Mackay, 1992; Graves, 2011; Louizos and Welling, 2016; Blundell et al., 2015).

Gaussian Processes. As alluded earlier, an ideal modeling approach in the era of big data should possess not only a powerful fitting capability but also a firm mechanism to determine predictive uncertainty. Bayesian nonparametric approaches are ideal candidates due to their advantages over flexibility and calibrated predictive uncertainty. The philosophies and motivations of this area have been well discussed by a number of authors (Hjort et al., 2010; Ghosh and Ramamoorthi, 2011; Ghahramani, 2013). Gaussian Processes (GPs) (Rasmussen and Williams, 2006) are an attractive way of doing non-parametric Bayesian modeling. A Gaussian Process is a collection of

random variables indexed by a variable in the input domain, such that every subset of those random variables has a multivariate normal distribution. Thanks to the properties of the multivariate normal distribution, given observations, GPs are able to make inferences as well as predictive uncertainties with a firm mathematical background. In addition to providing uncertainty in predictions, there are also compelling reasons to use GPs, such as the GPs can represent a rich family of functions; also, GPs are protected from overfitting with an appropriate prior on hyperparameters. In practice, GPs achieve state-of-the-art results in a wide spectrum of applications including robotics (Ko and Fox, 2008; Deisenroth and Rasmussen, 2011), geostatistics (Diggle and Ribeiro, 2007), numerics (Briol et al., 2015), active sensing (Guestrin et al., 2005) and optimization (Snoek et al., 2012).

Deep Gaussian Processes. A shallow GP is defined by a mean and covariance/kernel function. Kernel functions hold a crucial role as it not only encodes our assumptions as well as the desired flexibility into the functions we wish to learn. Thus, enhancing the expressiveness of kernel functions are able to boost the GPs' power. A Deep Gaussian Process DGP (Damianou and Lawrence, 2013) which is a hierarchical composition of multiple GPs, comes to a rescue of the limitation of the representational power of a single-layer GP. DGPs is more flexible than a standard GP, just as deep neural networks are more powerful than a Multilayer Perceptron with one hidden layer. In contrast to models constructed by with a highly parameterized functional form, DGPs learn a hierarchical representation with very few hyperparameters to optimize.

1.2 Extensions and Open Problems

In this section, I introduce the extensions and open problems of (Deep) Gaussian Processes which will appear in the dissertation.

Combination of Neural Networks and Gaussian Processes. In 1996, Neal Neal (1996) showed that Bayesian Neural Networks with infinitely many hidden units converged to Gaussian Processes (GPs) with a particular kernel function. Speaking theoretically, Gaussian Processes were viewed as an interpretable alternative to neural networks. However, in practice, the power of GPs are restricted by the limitations of the kernel function. By contrast, neural networks are able to automatically discover meaningful representations in high-dimensional data by learning multiple layers of highly adaptive basis functions MacKay (1998); Hinton et al. (2006); Bengio (2009).

Despite the impressive expressiveness, neural networks access predictive uncertainties via sampling using approaches Welling and Teh (2011); Gal and Ghahramani (2016a); Lakshminarayanan et al. (2017). Unlike neural networks, GPs directly capture predictive uncertainties with a firm mathematical background. Another advantage of GPs over neural networks is that the prior knowledge about the properties of mapping function, e.g. smoothness, differentiability or periodicity, can be added by specifying an appropriate kernel function.

As neural networks and GPs have particular strengths, the question of what the best paradigm, e.g. kernel methods in general (Gaussian Processes in particular) and neural networks) is become irrational. Instead, it is more sensible to think about the idea of combining the advantages of each approach. There are several works about the combinations of convolutional neural networks and GPs on image recognition, e.g. substituting GPs for the last fully connected layers Bradshaw et al. (2017); Wilson et al. (2016) or introducing convolutions in the calculation of the covariance between images van der Wilk et al. (2017).

Evaluation of Predictive Uncertainty of Probabilistic Models. As alluded in the introduction section, predictive uncertainty can be evaluated by inspecting the calibration and out-of-distribution samples. The majority of works accessing predictive uncertainty on NN involve with Bayesian formalism Mackay (1992); Graves (2011); Louizos and Welling (2016); Blundell et al. (2015). Along a similar vein, combining CNN and GP is an intuitive way to add probabilistic flavor to CNN Bradshaw et al. (2017); Wilson et al. (2016); van der Wilk et al. (2017). Intuitively, the motivation to impose these Bayesian treatments into neural networks is to do a better quantification of uncertainty compared to plain neural networks. Nevertheless, analyzing Bayesian Neural Networks and the combination of neural networks and GPs on predictive uncertainty has not been conducted carefully.

Inducing point-based approximation. GPs Rasmussen and Williams (2006) are well-known because of the predictive uncertainties with a firm mathematical background. Despite being able to underpin a range of algorithms for supervised and unsupervised learning, the application of GPs is hindered to the large-scale problems due to the burden of computational and storage cost. Assuming that the input dimensionality D is significantly less than the number of observations N, GPs require the complexities of $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ for computation and storage. These costs are sourced from linear algebraic operation with the $N \times N$ kernel matrix. To improve the scalability

of GPs, we must employ a technique accelerating the computation involving the kernel matrix. Almost works discussing the scalable GP have focused on the low-rank approximation of kernel matrix using inducing points (Lawrence et al., 2002; Seeger et al., 2003; Snelson and Ghahramani, 2005; Naish-Guzman and Holden, 2007; Titsias, 2009; Hensman et al., 2013; Wilson and Nickisch, 2015; Hensman et al., 2015a). Using M inducing points to obtain an approximation to the kernel matrix, the computational and storage costs are contracted to $\mathcal{O}(M^3)$ and $\mathcal{O}(M^2)$ respectively. It is obvious that inducing point-based approaches lead to a remarkable development on the scalability of GPs if M is significantly less than N.

Recently, it has been shown that it is possible to obtain an arbitrarily good approximation for a certain class of GP models (i.e. conjugate likelihoods, concentrated distribution for the training data) with M growing more slowly than N. However, the general case remains elusive and it is still possible that the required value for M may exceed a certain computational budget. To employ a large number of inducing points without exploding the computational cost, these inducing inputs are arranged into a structure such that the resulting kernel matrix allows for the application of fast linear algebra, and the entries of the kernel matrix evaluated at the training inputs are approximated through interpolation via sparse matrices. A well-known example for this line of work was introduced by Wilson et al Wilson and Nickisch (2015), namely Kernel Interpolation for Scalable Structured GPs (KISS-GP). The applicability of KISS-GPon higher-dimensional problems has been addressed in Wilson et al. (2015) by means of low-dimensional projections. A more recent extension allows for a constant-time variance prediction using Lanczos methods Pleiss et al. (2018). The limitation of these approaches is that inducing inputs must abide by the Kronecker structure due to computational acceleration. This leads to the partial restriction on the freedom of the optimization of inducing inputs.

1.3 Outline and Contributions of Thesis

The content of this thesis is organized as follows:

- **Chapter 2** starts with a brief introduction to Gaussian Processes (GPs). We also investigate state-of-the-art techniques for dealing with the notorious limitation of GPs on time and storage complexity as well as the flexibility of kernel function. In this text, these approaches is grouped into

three main categories of approximations, namely inducing point-based approximations, structure exploiting approximations, random feature-based approximations are discussed. This chapter is intended to equip the reader with the background knowledge required for apprehending the underlying concepts presented in this thesis, and clarify how our contributions fit within the landscape of existing research on Gaussian process inference;

- **Chapter 3** covers the first primary contribution of this thesis. The study expresses a thorough investigation of the calibration properties of Bayesian Convolutional Neural Networks (CNNs) . Along a similar vein, independently of the works on Bayesian CNNs, there are other attempts to impose a probabilistic formalism to CNNs by integrating CNNs with GPs. Previous work on combining CNNs with GPs has been developed under the assumption that the predictive probabilities of these models are well-calibrated. We show that, in fact, current combinations of CNNs and GPs are miscalibrated. We propose a novel combination that considerably outperforms previous approaches to this aspect, while achieving state-of-the-art performance on image classification tasks.

- As alluded earlier, inducing point-based idea are a well-known approach to mitigate the computational bottleneck of GPs in the large-scale problems. However, this solution still suffers cubic time complexity to the number of inducing points. Wilson et al Wilson and Nickisch (2015) propose to employ the Kronecker structure on inducing inputs to accelerate the approximation of covariance matrices. The trick also accompanies with significant restrictions on inducing inputs. Besides, the approach only performs well on low-dimensional datasets (Wilson and Nickisch, 2015). In **Chapter 4**, we address one limitation of sparse GPs, which is due to the challenge in dealing with a large number of inducing variables without imposing a special structure on the inducing inputs. In particular, we introduce a novel hierarchical prior, which imposes sparsity on the set of inducing variables. The study enables the possibility to use sparse GPs using a large number of inducing points without incurring a prohibitive computational cost.

- Finally, in **Chapter 5**, we summarize the contributions presented in this thesis. We conclude the thesis by a discussion to an outlook on possible extensions and future work.

2

Gaussian Processes for Big Data

Increasing the scalability and representational power of models without compromising performance is a core problem in machine learning. As emphasized in the introduction to this dissertation, the scalability of Gaussian Processes to training size and dimensionality is significantly limited by algebraic operations, which discourages their application to datasets having more than a few thousands of examples or high-dimensional covariates. Additionally, the flexibility of Gaussian Processes is possibly weakened by the need to choose a kernel functions, which might lead to difficulties in learning the intricate patterns concealed in the data. This chapter is a literature review on the developments of GPs in both aspects, which involve the major contributions of the thesis.

2.1 Overview

Gaussian Processes (henceforth GPs) which are powerful non-parametric Bayesian models can yield sensible predictions with a small number of available observations. However, it is notorious that GPs suffer from high complexity in terms of both computation and storage with respect to training size N, i.e. $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ respectively, so they not the primary choice in datasets with a massive number of data points. To broaden the application of GPs to larger datasets, there is plenty of ideas in the literature that have been proposed and analyzed. According to the groupings mentioned in (Liu et al., 2018b), these approaches are categorized into global and local approximations. While the former approximate the full kernel matrix by a global distillation, the latter abide to the divide-and-conquer concept and make predictions using a local subset of training data. We further split global approximations into subcategories: Inducing Point-Based Approximation and Random Feature-Based Approximation.

2.2 Gaussian Processes

As alluded earlier, a modeling approach in the era of big data should possess not only a powerful fitting capability but also a firm mechanism on predictive uncertainty. Bayesian nonparametrics is obviously an ideal candidate as it offers flexibility as well as calibrated predictive uncertainties. The philosophy and motivation of this area have been well discussed by a number of authors Ghosh and Ramamoorthi (2011); Hjort et al. (2010); Ghahramani (2013). Gaussian Process (GPs) Rasmussen and Williams (2006) are an attractive way of doing non-parametric Bayesian modeling in supervised learning problems. Firstly, I succinctly introduce Gaussian Processes Regression (GPR) which is the simplest way to describe GPs.

2.2.1 Gaussian Processes for Regression

Given a dataset $\mathcal{D}$ of N examples, $\mathcal{D} = \{(\mathbf{x}_n,\ y_n) \mid n = 1, \ldots, N\}$, where $\mathbf{x}_n$ denotes the n-th input vector (covariates) and y_n denotes the n-th scalar output or target; the column vector inputs for all N cases are aggregated in the $D \times N$ design matrix $\mathbf{X}$, and the outputs are collected in the vector $\mathbf{y}$, so we can write $\mathcal{D} = (\mathbf{X}, \mathbf{y})$. We would like to specify a function y representing the correlation between inputs and its targets, i.e. $y_n = y(\mathbf{x}_n)$. From a generative perspective, the observable labels $y(\mathbf{x}_n)$ are modeled via an appropriate conditional likelihood $p(y(\mathbf{x}_n) \mid f(\mathbf{x}_n))$, where f is the latent function which can also be perceived as the intermediate representation of function y. In regression, the conditional likelihood is intuitively often assumed to be a Gaussian with mean of $\mathbf{f}$ and variance of σ_n^2, i.e. $p(\mathbf{y} \mid \mathbf{f}, \sigma_n^2) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma_n^2 \mathbf{I})$. In general, the function f can be viewed as a column vector $\mathbf{f}$, i.e. $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^T$, where $f(\mathbf{x}_n)$ is latent values at input $\mathbf{x}_n$. Formally, GPs are formally defined as a prior over latent function f, but with the view of latent function f as a finite-dimensional vector, GPs turns out a multivariate Gaussian distributions over $\mathbf{f}$. A GPs prior is fully specified by its mean $m(\mathbf{x} \mid \boldsymbol{\zeta})$ and covariances which are determined by a predefined kernel functions $k(\mathbf{x}_i, \mathbf{x}_j \mid \boldsymbol{\theta})$. Here, $\boldsymbol{\zeta}$ and $\boldsymbol{\theta}$ are parameters of mean function m and kernel function k respectively. The GPs prior over latent values $\mathbf{f}$ given $\boldsymbol{\zeta}$ and $\boldsymbol{\theta}$ is as follows:

$$p(\mathbf{f} \mid \mathbf{X}, \boldsymbol{\zeta}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{m}_\mathbf{X}, \mathbf{K}_\mathbf{X}). \tag{2.1}$$

where $\mathbf{m}_\mathbf{X}$ are column N-dimensional vector containing mean values at N covariates, i.e. $\mathbf{m}_\mathbf{X} = [m(\mathbf{x}_1 \mid \boldsymbol{\zeta}), \ldots, m(\mathbf{x}_N \mid \boldsymbol{\zeta})]^T$; and $\mathbf{K}_\mathbf{X}$ is a $N \times N$ symmetric and positive semi-definite matrix representing the correlation between latent random variables each other, $[\mathbf{K}_\mathbf{X}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j \mid \boldsymbol{\theta})$.

Hyper-parameter optimization. For convenience sake, we introduce $\boldsymbol{\psi}$ the set of all parameters involving mean function's parameters $\boldsymbol{\zeta}$, kernel parameters' $\boldsymbol{\theta}$ and variance of likelihood σ_n^2, i.e. $\boldsymbol{\psi} = (\sigma_n^2, \boldsymbol{\zeta}, \boldsymbol{\theta})$. Given dataset $\mathcal{D}$, Gaussian Processes Regressors are fitted to $\mathcal{D}$ by optimizing hyper-parameter $\boldsymbol{\psi}$ using the logarithm marginal likelihood function, $\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\psi})$. In general, the marginal likelihood can be found by marginalizing over latent random variables $\mathbf{f}$.

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\psi}) = \int p\left(\mathbf{y} \mid \mathbf{f}, \sigma_n^2\right) p(\mathbf{f} \mid \mathbf{X}, \boldsymbol{\zeta}, \boldsymbol{\theta})\, d\mathbf{f}. \tag{2.2}$$

Thanks to the Gaussian likelihood $p(\mathbf{y} \mid \mathbf{f}, \sigma_n^2)$, we can derive an analytic form for the marginal likelihood as the Gaussian likelihood and Gaussian prior are conjugate to each other.

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\psi}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{m}_{\mathbf{X}}, \mathbf{K}_{\mathbf{X}} + \sigma_n^2 \mathbf{I}\right). \tag{2.3}$$

Setting $\mathbf{K}_{\sigma_n^2}$ as $\mathbf{K}_{\mathbf{X}} + \sigma_n^2 \mathbf{I}$, the logarithm marginal likelihood is written as:

$$\log\left[p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\psi})\right] = -\frac{1}{2}\log\left|\mathbf{K}_{\sigma_n^2}\right| - \frac{1}{2}\left(\mathbf{y} - \mathbf{m}_{\mathbf{X}}\right)^T \mathbf{K}_{\sigma_n^2}^{-1}\left(\mathbf{y} - \mathbf{m}_{\mathbf{X}}\right) - \frac{N}{2}\log 2\pi. \tag{2.4}$$

The quadratic form appearing in this expression corresponds to the model fit term of the GPR, advocating parameter settings that fit the data well. In contrast, the log determinant term penalizes overly complex models that are characterized by kernel matrices which are diagonally dominant, indicating little interaction between observations. It follows that the optimal parameters $\boldsymbol{\psi}_{OPT}$ are identified by maximizing this objective function using iterative gradient ascent.

Prediction. Generally, GPs governs the distribution of a finite-dimensional vector including latent values at a set of covariates using a multivariate normal distribution. Therefore, the joint distribution of training latent values, $\mathbf{f}$, and the testing latent values, $\mathbf{f}_*$, according to the GP prior is:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_{\mathbf{X}} \\ \mathbf{m}_{\mathbf{X}_*} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{X}_*} \\ \mathbf{K}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{K}_{\mathbf{X}_*} \end{bmatrix}\right) \tag{2.5}$$

If $\mathbf{X}$ and $\mathbf{X}_*$ include N training points and N_* testing points, respectively, then $\mathbf{m}_{\mathbf{X}}$ and $\mathbf{m}_{\mathbf{X}_*}$ contain N and N_* values of the mean function at $\mathbf{X}$ and $\mathbf{X}_*$; and $\mathbf{K}(\mathbf{X}, \mathbf{X}_*)$ denotes the $N \times N_*$ matrix of the covariances evaluated at all pairs of training and testing points, and similarly for the other covariance matrices.

Remind that, in regression, the likelihood of observable targets given training latent values are intuitively assumed to be a Gaussian with the variance of σ_n^2, $p(y_n \mid f_n) = \mathcal{N}(y_n \mid f_n, \sigma_n^2)$. It means that the functions for observable targets can be modeled as a noisy version of latent function f a Gaussian noise with variance of σ_n^2, $y(\mathbf{x}_n) = f(\mathbf{x}_n) + \varepsilon$, where ε follows $\mathcal{N}(\varepsilon \mid 0, \sigma_n^2)$. Assuming additive independent identically distributed Gaussian noise with variance σ_n^2, the prior on the noisy observations becomes:

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{y} \mid \mathbf{m}_{\mathbf{X}}, \mathbf{K}_{\mathbf{X}} + \sigma_n^2 \mathbf{I}\right). \tag{2.6}$$

We can write the joint distribution of the observed target values and the function values at the test locations under prior as:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_{\mathbf{X}} \\ \mathbf{m}_{\mathbf{X}_*} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathbf{X}} + \sigma_n^2 \mathbf{I} & \mathbf{K}_{\mathbf{X},\mathbf{X}_*} \\ \mathbf{K}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{K}_{\mathbf{X}_*} \end{bmatrix}\right) \tag{2.7}$$

To get the posterior distribution over function, we need to restrict this joint prior distribution to contain only those functions which agree with the observed data points. By virtue of the nice properties of the multivariate normal distribution, the operation of eliminating those violating the available observations is extremely simple, corresponding to conditioning the joint Gaussian prior distribution on the observations to give:

$$\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\psi} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{f}_*}, \boldsymbol{\Sigma}_{\mathbf{f}_*}\right), \text{ where} \tag{2.8}$$

$$\boldsymbol{\mu}_{\mathbf{f}_*} = \mathbf{m}_{\mathbf{X}_*} + \mathbf{K}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{K}_{\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1}(\mathbf{f} - \mathbf{m}_{\mathbf{X}}), \text{ and,} \tag{2.9}$$

$$\boldsymbol{\Sigma}_{\mathbf{f}_*} = \mathbf{K}_{\mathbf{X}_*} - \mathbf{K}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{K}_{\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1}\mathbf{K}_{\mathbf{X},\mathbf{X}_*}. \tag{2.10}$$

Once again, thanks to Gaussian likelihood with noise variance of σ_n^2, the predictive distribution $p(\mathbf{y}_* \mid \mathbf{X}, \mathbf{y}, \boldsymbol{\psi})$ turns out:

$$\begin{aligned} p(\mathbf{y}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\psi}) &= \int p(\mathbf{y}_* \mid \mathbf{f}_*)\, p(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\psi})\, d\mathbf{f}_* \\ &= \mathcal{N}\left(\mathbf{y}_* \mid \boldsymbol{\mu}_{\mathbf{f}_*}, \boldsymbol{\Sigma}_{\mathbf{f}_*} + \sigma_n^2 \mathbf{I}\right) \end{aligned} \tag{2.11}$$

2.2.2 Covariance function

In GPs or any kernel machine learning methods, the notion of similarity between data points is crucial as the predictions are made based upon these similarities. Under the Gaussian process view, a covariance matrix specified by a kernel function defines nearness or similarity between latent random variables by using inputs. Therefore, it is able to encode our assumptions about

the function which we wish to learn through. It is uncertain whether an arbitrary matrix of input pair $\mathbf{x}_i$ and $\mathbf{x}_j$ will be a valid kernel function or not. The first purpose of the section is to show the properties and construction of a valid covariance function. In addition, examples of some commonly-used covariance functions in this dissertation are also given.

Construction and properties. The covariance matrix of the is constructed from a kernel function k of an input pair. Consider a GPs for the sequence of N latent values, the dimensionality of the covariance matrix of GPs is $N \times N$, and the element at i-th row and j-column of the covariance matrix is kernel function values of $\mathbf{x}_i$ and $\mathbf{x}_j$, $k(\mathbf{x}_i, \mathbf{x}_j)$. In general, the kinds of kernel function for all examples $\mathbf{x}_i$ and $\mathbf{x}_j$ in an input space $\mathcal{X} \subset \mathbb{R}^D$:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle. \tag{2.12}$$

, where ϕ is a non-linear (or linear) map from the input space $\mathcal{X}$ to the feature space $\mathcal{F}$, and $\langle ., . \rangle$ is an inner product. Due to being computed by the inner product, a kernel function must be symmetric and also satisfy the Cauchy-Schwartz inequality:

$$k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i), \text{ and } k^2(\mathbf{x}_i, \mathbf{x}_j) \leq k(\mathbf{x}_i, \mathbf{x}_i) k(\mathbf{x}_j, \mathbf{x}_j). \tag{2.13}$$

Practically, the kernel function k is usually specified directly, thus implicitly defining the map ϕ and the feature space $\mathcal{F}$. Therefore, a kernel function is stated to be valid if it guarantees the existence of the feature space. Mercer Mercer (1909) showed that a necessary and sufficient condition for a symmetric function $k(., .)$ to be a kernel is that it be positive definite. This means that for any set of $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and any set of real numbers $\lambda_1, \ldots, \lambda_N$, the function k must satisfy:

$$\forall \mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}, \forall \lambda_1, \ldots, \lambda_N \in \mathbb{R}, \sum_{i,j=1}^{N} \lambda_i \lambda_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \tag{2.14}$$

In summary, a symmetric positive definite function constructs a valid covariance matrix in kernel methods. As the positive definiteness possesses pleasant algebraic properties, a new kernel can be created from existing valid kernels. Introducing a_1 and a_2 are positive real numbers, and k_1 and k_2 are valid kernels, a new kernel can be manipulated using a weighted summation or multiplication:

$$k(\mathbf{x}_i, \mathbf{x}_j) = a_1 k_1(\mathbf{x}_i, \mathbf{x}_j) + a_2 k_2(\mathbf{x}_i, \mathbf{x}_j). \tag{2.15}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = k_1(\mathbf{x}_i, \mathbf{x}_j) k_2(\mathbf{x}_i, \mathbf{x}_j). \tag{2.16}$$

Stationary covariance function. A stationary covariance function of $\mathbf{x}_i$ and $\mathbf{x}_j$ only depends on Euclidean distance of $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e. $k(\mathbf{x}_i, \mathbf{x}_j) = k_S(r)$, where $r = \sqrt{|| \mathbf{x}_i - \mathbf{x}_j ||_2^2}$. Thus, it is invariant to translations in the input space. This kind of kernel are commonly-used because, intuitively, it is a basic similarity assumption that points with inputs x which are close are likely to have similar target values y, and thus training points that are near to a test point should be informative about the prediction at that point. Next, we mention two commonly-used isotropic kernel functions. The covariance functions are given in a normalized form where $k(0) = 1$; we can multiply k by a (positive) constant σ_f^2 to get any desired process variance.

- Squared Exponential Covariance Function.

  The square exponential function or Radial Basis Function (RBF) kernel has the form:

$$k_{RBF}(r) = \exp\left(-\frac{r^2}{2l^2}\right). \tag{2.17}$$

  , with positive parameter l defines the characteristic length-scale which indicating the complexity of underlying latent functions.

- The Matérn Covariance Function.

  The Matérn class of covariance functions is given by

$$k_{\text{Matérn}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}r}{l}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}r}{l}\right). \tag{2.18}$$

  , with positive parameters ν and l, and K_ν is a modified Bessel function Abramowitz (1974). The most interesting cases of Matérn class for machine learning are $\nu = 3/2$ and $\nu = 5/2$, for which

$$k_{\text{Matérn } 3/2}(r) = \left(1 + \frac{\sqrt{3}r}{l}\right)\exp\left(\frac{\sqrt{3}r}{l}\right), \tag{2.19}$$

$$k_{\text{Matérn } 5/2}(r) = \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right)\exp\left(-\frac{\sqrt{5}r}{l}\right), \tag{2.20}$$

Automatic Relevance Determination Kernel. The kernel functions mentioned above are called isotropic where the flexibility of kernel function is indicated by a lengthscale parameter, l. To enhance the flexibility of kernel function, we augment D length-scale parameters, $l_1, \ldots, l_D$ accompanying with D input dimensionality. It turns out that the term r/l in the isotropic

kernel is replaced using a quadratic form. For example, the RBF kernel can be rewritten as:

$$k_{\text{RBF}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{1}{2}\left(\mathbf{x}_i - \mathbf{x}_j\right)^T \Lambda^{-1}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right). \tag{2.21}$$

, where $\Lambda = \text{Diag}\left[l_1^2, \ldots, l_D^2\right]^T$. This interpretation of the lengthscales allows for automatic relevance determination whereby relevant features in the data are weighted by their corresponding lengthscale parameter. This can also be seen as an implicit form of feature selection (MacKay, 1991).

2.2.3 Non-Gaussian Likelihoods

Recall that in GP regression the Gaussian likelihood $p\left(\mathbf{y} \mid \mathbf{f}\right)$ is conjugate to the Gaussian prior $p\left(\mathbf{f}\right)$. Thus, it is possible to calculate the marginal likelihood and carrying out inference in GP regression analytically. In contrast, these calculations are analytically intractable in GP models with a non-Gaussian likelihood. There is a plethora of approaches to deal with the problem, including the Laplace approximation method (Williams and Barber, 1998), expectation propagation (Minka, 2001), sparse approximation employing online learning schemes (Lawrence et al., 2002; Csató and Opper, 2002) and methods attempting to characterize the full posterior (Murray et al., 2010; Filippone et al., 2013; Hensman et al., 2015b). As the prerequisite backgrounds for proposed models which will be introduced in the next chapters do not significantly depend on the techniques of approximating posterior with non-Gaussian likelihood, the discussion about the non-Gaussian likelihood or GPs classification will not be provided in this manuscript.

2.2.4 Limitations of Gaussian Processes

Scalability. Theoretically, GPs is an ideal approach for the supervised scenario in the era of big data. However, the scalability of GPs is limited on small datasets including a few thousands of data points due to linear algebraic operations requiring large computational complexity. Having considered the optimization of GPs hyper-parameters $\boldsymbol{\psi}$, the problem of GPs scalability is revealed. As alluded in section 2.2.1, the process of fitting GPs regressors given a dataset can be done by using a gradient-based method with the target function of marginal likelihood $p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\psi}, \mathbf{y}\right)$. Take GPs regression with zero mean prior as an example, the gradients of marginal likelihood with respect to parameter ψ_i is computed as:

$$\frac{\partial \log\left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}\right)\right]}{\psi_i} = -\frac{1}{2}\text{Tr}\left(\mathbf{K}_{\sigma_n^2}^{-1}\frac{\partial \mathbf{K}_{\sigma_n^2}}{\partial \psi_i}\right) + \frac{1}{2}\mathbf{y}^T\mathbf{K}_{\sigma_n^2}^{-1}\frac{\partial \mathbf{K}_{\sigma_n^2}}{\partial \psi_i}\mathbf{K}_{\sigma_n^2}^{-1}\mathbf{y}. \tag{2.22}$$

The computation of gradients involves with solving the linear system, i.e. $\mathbf{K}_{\sigma_n^2}^{-1}\mathbf{y}$ where $\mathbf{K}_{\sigma_n^2}$ is $N \times N$ covariance matrix with additive noise and $\mathbf{y}$ is N-dimensional column vector of outputs, where N is the number of data points. Practically, this linear system is solved by using Cholesky decomposition to factorize the symmetric positive definite kernel matrix $\mathbf{K}_{\sigma_n^2}$ into $\mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a lower triangular matrix. Generally, factorization with Cholesky decomposition necessitates $\mathcal{O}(N^3)$ operations. The calculation of the trace terms appearing in gradient formula also need $\mathcal{O}(N^3)$ operations. In the progress of computing the gradients, the lower triangular matrix $\mathbf{L}$ must be cached. Therefore, the storage cost of the training phase is $\mathcal{O}(N^2)$.

Besides the cubic complexity in the training phase, the computational cost of GPs inference also depends on the training size. On inspection of the predictive distribution given from equation 2.8 to 2.10, we can observe that evaluating this expression also involves the inversion of $N \times N$ kernel matrix. Theoretically, the computational cost for GP inference is also $\mathcal{O}(N^3)$. However, in practice, the inversion of $\mathbf{K}_{\sigma_n^2}$ and the vector which is the multiplication of $\mathbf{K}_{\sigma_n^2}^{-1}$ and $\mathbf{y} - \mathbf{m}_\mathbf{X}$ can be recorded after the training phase. Therefore, the computational costs of predictive mean and predictive variance at an unseen data point are $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$. As discussed, the likelihood mapping the latent values to observation is not obligated to be a Gaussian, as the case of classification. Under these conditions, the computation of the marginal likelihood as well as inference is no longer analytic, and further approximations are required, and computational budgets required in these case is identical to GP regression. Due to the dependence of computational complexity on training size, GPs are hindered to large-scale problems.

To strengthen the scalability, while retaining the desired prediction quality, a large number of scalable GPs have been proposed. According to (Liu et al., 2018b), these scalable approaches are sorted into two main categories: local and global approximation. Local approximations arising from the divide-and-conquer concept focus on the local area of input spaces (Gramacy and Lee, 2007; Yuksel et al., 2012b; Masoudnia and Ebrahimpour, 2014; Rasmussen and Ghahramani, 2002; Sun and Xu, 2011; Hinton, 2002; Deisenroth and Ng, 2015; Rullière et al., 2016; Liu et al., 2018a). Whereas global approximations replace kernel matrix $\mathbf{K}_\mathbf{X}$ by a compact representation reducing the burden of computation. The substitution is done through global distillation which can be accomplished by several ways, e.g. use a small subset of training data (Chalupka et al., 2013), or remove uncorrelated entries in $\mathbf{K}_\mathbf{X}$ using sparse kernel (Gneiting, 2002), or employ low-rank representation (Nyström approximation) (Hensman et al., 2013; Quiñonero Candela and Rasmussen, 2005;

Titsias, 2009; Wilson and Nickisch, 2015).

Representational power. Kernel functions hold a crucial role as it not only encodes our assumptions as well as the desired flexibility into the functions we wish to learn. Concerning the representational capability, kernel-based methods possibly lose their power as very limited kernels such as RBF kernel sharing a single length-scale across input are overused, e.g. in some gp-based approaches and, especially in Support Vector Machine (SVM) . Having been encouraged by the achievement of deep architectures, there have been several attempts to build kernel-based method that mimic deep neural networks, for example, multilayer ARC-COSINE kernel (Cho and Saul, 2009) which is built by successive kernel compositions, and kernel function at each layer are defined via an integral representation, or convolutional multilayer kernels (Mairal et al., 2014) which are built by concatenations of convolutional layers, and the compact representation of the kernel are learned in a data-dependent manner. Another approach to enhance the flexibility of kernel methods is to use its deep architecture, e.g. Deep GPs (Damianou and Lawrence, 2013; Salimbeni and Deisenroth, 2017; Cutajar et al., 2017).

2.3 Inducing Point Approximations

2.3.1 Prior approximation

Main idea. As mentioned in section 2.2.4, the computational bottleneck of Gaussian Processes (GPs) stems from the algebraic operation of the full kernel matrix that appears in the prior distribution. Intuitively, the idea of employing the approximations to these true priors accelerating the computations come to a rescue for the problem of scalability. In this approach, the joint prior $p(\mathbf{f}_*, \mathbf{f})$ is modified in ways that reduces the computational cost. Here, $\mathbf{f}_*$ and $\mathbf{f}$ are the latent values at training points $\mathbf{X}$ and testing points $\mathbf{X}_*$ respectively. For clarity, it is useful to derive the exact expression for the joint prior before discussing about the particular approaches employing the idea. Without loss of generality, the mean of all priors is set to zero. Introducing the auxiliary random variables $\mathbf{u}$, which are latent values at inducing inputs $\mathbf{Z}$, the joint prior $p(\mathbf{f}_*, \mathbf{f})$ is expressed by marginalizing out $\mathbf{u}$ from the joint prior $p(\mathbf{f}_*, \mathbf{f}, \mathbf{u})$.

$$p(\mathbf{f}_*, \mathbf{f}) = \int p(\mathbf{f}_*, \mathbf{f}, \mathbf{u})\, d\mathbf{u} = \int p(\mathbf{f}_*, \mathbf{f} \mid \mathbf{u})\, p(\mathbf{u})\, d\mathbf{u} \tag{2.23}$$

Due to the consistency of GPs, all probabilistic components appearing in equation 2.23, i.e. the joint prior $p(\mathbf{f}_*, \mathbf{f}, \mathbf{u})$ and the conditional prior $p(\mathbf{f}_*, \mathbf{f} \mid \mathbf{u})$

and the prior $p(\mathbf{u})$ are Gaussian densities. Introducing $\hat{\mathbf{f}}$ as the general latent values for both training and testing points, we can rewrite the joint prior as:

$$\text{Joint prior: } p(\mathbf{f}_*, \mathbf{f}) = p\left(\hat{\mathbf{f}}\right) = \int p\left(\hat{\mathbf{f}}, \mathbf{u}\right) d\mathbf{u} = \int p\left(\hat{\mathbf{f}} \mid \mathbf{u}\right) p(\mathbf{u})\, d\mathbf{u} \quad (2.24)$$

$$\text{Prior: } p(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K}_{\mathbf{Z},\mathbf{Z}}) \quad (2.25)$$

$$\text{Conditional: } p\left(\hat{\mathbf{f}} \mid \mathbf{u}\right) = \mathcal{N}\left(\hat{\mathbf{f}} \mid \mathbf{K}_{\hat{\mathbf{X}},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{u}, \mathbf{K}_{\hat{\mathbf{X}},\hat{\mathbf{X}}} - \mathbf{Q}_{\hat{\mathbf{X}},\hat{\mathbf{X}}}\right) \quad (2.26)$$

Here, $\hat{\mathbf{X}}$ generally indicates training inputs $\mathbf{X}$ and testing inputs $\mathbf{X}_*$. Assuming that $\mathbf{A}$ and $\mathbf{B}$ are the matrices constructed by concatenating covariates likewise $\mathbf{X}$ and $\mathbf{X}_*$, we define $\mathbf{K}_{\mathbf{A},\mathbf{B}}$ as a cross covariance matrix whose element in the i, j position is the covariance between the i-th covariate in $\mathbf{A}$ and j-th covariate in $\mathbf{B}$. We also introduce the shorthand notation $\mathbf{Q}_{\mathbf{A},\mathbf{B}} = \mathbf{K}_{\mathbf{A},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{B}}$ which can be seen as an approximation to $\mathbf{K}_{\mathbf{A},\mathbf{B}}$ using inducing inputs $\mathbf{Z}$. For simplicity, we use the Gaussian likelihood $p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma_n^2\mathbf{I})$. The predictive latent distributions $p(\mathbf{f}_* \mid \mathbf{y})$ can be written in a closed-form using Gaussian density:

$$\begin{aligned} p(\mathbf{f}_* \mid \mathbf{y}) = \mathcal{N}\big(\mathbf{f}_* \mid & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}, \\ & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{K}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}}\right)^{-1}\mathbf{K}_{\mathbf{X},\mathbf{X}_*} + \sigma_n^2\mathbf{I}\big). \end{aligned} \quad (2.27)$$

By assuming that $\mathbf{u}$ captures all correlation between $\mathbf{f}_*$ and $\mathbf{f}$, i.e. $\mathbf{f}_*$ and $\mathbf{f}$ are independent given $\mathbf{u}$, we can approximate $p(\mathbf{f}_*, \mathbf{f} \mid \mathbf{u})$ by separating training latent values $\mathbf{f}$ and testing latent values $\mathbf{f}_*$:

$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) = \int q(\mathbf{f}_* \mid \mathbf{u})\, q(\mathbf{f} \mid \mathbf{u})\, p(\mathbf{u})\, d\mathbf{u}. \quad (2.28)$$

Following the unifying view mentioned by Quiñonero Candela and Rasmussen (2005), a particular algorithm complying with the idea of prior approximation corresponds to different additional assumptions about the two approximate inducing conditionals $q(\mathbf{f} \mid \mathbf{u})$ and $q(\mathbf{f}_* \mid \mathbf{u})$ appearing in the approximation defined in 2.28. The method PIC (Snelson and Ghahramani, 2007) mentioned at the end of the section is also an extension of the idea by using another way to approximate the joint prior $p(\mathbf{f}_*, \mathbf{f})$.

Subset of Data. The most straightforward approach to reduce the computational burden of GPs, which stems from the inverse of the kernel matrix $\mathbf{K}_{\mathbf{X}}$, is to work on subsets of the data (henceforth SOD), $\mathcal{D}_{\text{SOD}}$ for the whole training points, $\mathcal{D}$, i.e. simply speaking, we use $\mathbf{K}_{\mathbf{X}_{\text{SOD}}}$ instead of $\mathbf{K}_{\mathbf{X}}$. By restricting the number of data point M in $\mathbf{X}_{\text{SOD}}$ to be less than the total number of observations, N, the computational cost will decrease from $\mathcal{O}(N^3)$ to

$\mathcal{O}(M^3)$. In case $\mathbf{X}_{\text{SOD}}$ is specified in an appropriate manner, the approaches of SOD will produce reasonable predictive distributions. Otherwise and most often, SOD yields overconfident predictions. On the inspection of the selection of $\mathcal{D}_{\text{SOD}}$, one could, for example, randomly choose M data points, use clustering techniques to divide the training data to M subsets and then choose the centroids as representative for all the whole data sets, or employ online learning scheme with criteria based on information theoretic principles, i.e. differential entropy (Lawrence et al., 2002), to choose active data points sequentially.

Turning to the unifying view mentioned above, $\mathbf{u}$ and $\mathbf{f}$ are replaced by $\mathbf{f}_{sod}$ which are the latent values of subset input $\mathcal{D}_{\text{SOD}}$. SOD also uses the true testing conditional distribution instead of its approximation, i.e. $q(\mathbf{f}_* \mid \mathbf{u}) = p(\mathbf{f}_* \mid \mathbf{f}_{sod})$. The joint prior turns out to be:

$$p(\mathbf{f}_*, \mathbf{f}) \to p(\mathbf{f}_*, \mathbf{f}_{sod}) = \int p(\mathbf{f}_* \mid \mathbf{f}_{sod})\, p(\mathbf{f}_{sod}) \tag{2.29}$$

Subset of Regressors. According to the study on Subset of Regressors (SOR) of Silverman (1985) and Wahba et al. (1999), Smola and Bartlett (2001) have adjusted SOR for a sparse approximation to Gaussian Processes Regression. SOR assumes that there is a deterministic relationship between latent values, i.e. $\mathbf{f}_*$ and $\mathbf{f}$, and inducing variables $\mathbf{u}$. This correlation can be represented as a Gaussian distribution with zero covariance as follows:

$$q_{\text{SOR}}\left(\hat{\mathbf{f}} \mid \mathbf{u}\right) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{\hat{\mathbf{X}},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{u}, \mathbf{0}\right). \tag{2.30}$$

Substituting $q_{\text{SOR}}\left(\hat{\mathbf{f}} \mid \mathbf{u}\right)$ to the Equation 2.28, the approximated joint prior is:

$$q_{\text{SOR}}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f} \\ \mathbf{f}_*\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0} \\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{Q}_{\mathbf{X},\mathbf{X}} & \mathbf{Q}_{\mathbf{X},\mathbf{X}_*} \\ \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}_*}\end{bmatrix}\right) \tag{2.31}$$

From the approximated joint prior and the Gaussian likelihood, we can obtain the approximated predictive latent distribution:

$$\begin{aligned} q_{\text{SOR}}(\mathbf{f}_* \mid \mathbf{y}) = \mathcal{N}\big(\mathbf{f}_* \mid & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}, \\ & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{Q}_{\mathbf{X},\mathbf{X}_*}\big) \end{aligned} \tag{2.32}$$

Having observed the true predictive latent distributions $p(\mathbf{f}_* \mid \mathbf{y})$ defined in Equation 2.27, the approximated predictive latent distributions $q_{\text{SOR}}(\mathbf{f}_* \mid \mathbf{y})$ are identical with $p(\mathbf{f}_* \mid \mathbf{y})$, except that the covariance $\mathbf{K}$ has been substituted by $\mathbf{Q}$. Therefore, SOR approximation operates as an exact Gaussian Processes with the covariance matrix K_{SOR} defined by the kernel function

$k_{\text{SOR}}(\mathbf{x}_i, \mathbf{x}_j) = K_{\mathbf{x}_i,\mathbf{Z}} \mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1} \mathbf{K}_{\mathbf{Z},\mathbf{x}_j}$.

Deterministic Training Conditional. According to the analysis of Williams et al. (2002), SOR can yield negative predictive variances due to the approximation of the full covariance matrix using the Nyström method. In order to avoid these nonsensical predictive variances, Seeger and Williams (2003) proposed a novel sparse approximation to Gaussian Processes Regression. The approach mainly relies on a likelihood approximation, based on the projection of training latent values, i.e. $\mathbf{f} = \mathbf{K}_{\mathbf{X},\mathbf{Z}} \mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1} \mathbf{u}$. Due to the deterministic projection, this approach is called to Deterministic Training Conditional (DTC) .

$$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{f}, \sigma_n^2 \mathbf{I}\right) \approx q_{\text{DTC}}(\mathbf{y} \mid \mathbf{u}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{K}_{\mathbf{X},\mathbf{Z}} \mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1} \mathbf{u}, \sigma_n^2 \mathbf{I}\right) \quad (2.33)$$

This approach uses the point estimate to variational distribution over training latent value similarly to SOR, it remains to use the exact test conditional defined in 2.26.

$$q_{\text{DTC}}(\mathbf{f} \mid \mathbf{u}) = q_{\text{SOR}}(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{\mathbf{X},\mathbf{Z}} \mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1} \mathbf{u}, \mathbf{0}\right). \quad (2.34)$$

$$q_{\text{DTC}}(\mathbf{f}_* \mid \mathbf{u}) = p(\mathbf{f}_* \mid \mathbf{u}) = \mathcal{N}\left(\mathbf{f}_* \mid \mathbf{K}_{\mathbf{X}_*,\mathbf{Z}} \mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1} \mathbf{u}, \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}_*}\right). \quad (2.35)$$

Another difference between SOR and DTC is indicated in the joint prior. While SOR uses $\mathbf{Q}_{\mathbf{X},\mathbf{X}_*}$ to govern the relation between testing points, DTC use the exact full covariance matrix $\mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*}$.

$$q_{\text{DTC}}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f} \\ \mathbf{f}_*\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0} \\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{Q}_{\mathbf{X},\mathbf{X}} & \mathbf{Q}_{\mathbf{X},\mathbf{X}_*} \\ \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*}\end{bmatrix}\right). \quad (2.36)$$

The predictive distribution of DTC is similar to SOR, but $\mathbf{Q}_{\mathbf{X}_*,\mathbf{X}_*}$ is replaced by $\mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*}$:

$$\begin{aligned} q_{\text{DTC}}(\mathbf{f}_* \mid \mathbf{y}) = \mathcal{N}\big(\mathbf{f}_* \mid & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} \left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y}, \\ & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} \left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{Q}_{\mathbf{X},\mathbf{X}_*}\big) \end{aligned} \quad (2.37)$$

Fully Independent (Training) Conditional. The main limitation of sparse approximation to Gaussian Processes proposed before 2006 is that the active points are restricted to be a subset of training covariate. In additions, the fact that selections of active points are repeated in training progress causes non-smooth fluctuations in the marginal likelihood and its gradients, meaning that they cannot get smooth convergence. To circumvent the problem, Snelson and Ghahramani (2005) introduced an alternative sparse approximation to Gaussian Processes Regression which is called Sparse Gaussian Processes using Pseudo-inputs (SGPP) . This approach enables the joint optimization

of active locations and kernel hyper-parameters.

Integrating SGPP into the unifying framework, we can observe clearly the differences in the formalism between SGPP and SOR and DTC. While the likelihood variance of DTC is characterized by only the noise variance, the likelihood variance of SGPP also takes into account the residual difference between $\text{Diag}\left(K_{\mathbf{X},\mathbf{X}}\right)$ and $\text{Diag}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}}\right)$. SGPP assumes that the auxiliary variables $\mathbf{u}$ induces the relation of training latent variables $\mathbf{f}$. Due to this assumption, SGPP can be called Fully Independent Training Conditional (FITC) approximation. The approximation to the likelihood as well as the variational distribution of training and testing latent values given $\mathbf{u}$ also relies on the projection as in DTC, but the predictive variances are more sophisticated than DTC.

$$p\left(\mathbf{y} \mid \mathbf{f}\right) \approx q_{\text{FITC}}\left(\mathbf{y} \mid \mathbf{u}\right) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{K}_{\mathbf{X},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{u}, \text{Diag}\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}\right] + \sigma_n^2\mathbf{I}\right). \tag{2.38}$$

$$q_{\text{FITC}}\left(\mathbf{f} \mid \mathbf{u}\right) = \prod_{n=1}^{N} p\left(f_n \mid \mathbf{u}\right) = \prod_{n=1}^{N} \mathcal{N}\left(f_n \mid \mathbf{K}_{\mathbf{x}_n,\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{x}_n,\mathbf{x}_n} - \mathbf{Q}_{\mathbf{x}_n,\mathbf{x}_n}\right). \tag{2.39}$$

$$q_{\text{FITC}}\left(\mathbf{f}_* \mid \mathbf{u}\right) = p\left(\mathbf{f}_* \mid \mathbf{u}\right) = \mathcal{N}\left(\mathbf{f}_* \mid \mathbf{K}_{\mathbf{X}_*,\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}\right). \tag{2.40}$$

By introducing $\mathbf{A}$ as a square matrix, the operator $\text{Diag}\left(\mathbf{A}\right)$ constructs a diagonal matrix whose elements are taken from the diagonal line of $\mathbf{A}$. The approximation to joint prior $q_{\text{FITC}}\left(\mathbf{f}, \mathbf{f}_*\right)$ is similar to DTC, except for the covariance matrix governing the relation of training latent variables. While DTC uses $\mathbf{Q}_{\mathbf{X},\mathbf{X}}$ in $q_{\text{DTC}}\left(\mathbf{f}, \mathbf{f}_*\right)$ defined in the equation 2.36, FITC also uses $\mathbf{Q}_{\mathbf{X},\mathbf{X}}$ in the approximation to joint prior, but remain the true kernel value at diagonal elements.

$$q_{\text{FITC}}\left(\mathbf{f}, \mathbf{f}_*\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} & \mathbf{Q}_{\mathbf{X},\mathbf{X}_*}\\ \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*}\end{bmatrix}\right) \tag{2.41}$$

, where $\mathbf{\Lambda} = \text{Diag}\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}\right]$. From the joint prior defined in 2.41, the predictive distribution of FITC or SGPP turns out:

$$\begin{aligned} q_{\text{FITC}}\left(\mathbf{f}_* \mid \mathbf{y}\right) = \mathcal{N}\big(\mathbf{f}_* \mid & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y},\\ & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{Q}_{\mathbf{X},\mathbf{X}_*}\big). \end{aligned} \tag{2.42}$$

Observe the approximation to joint prior in FITC defined in the equation 2.41, we realize that the training and testing covariance are constructed heterogeneously. Therefore, the approximation FITC does not comply with the strict definition of GPs where the covariance for all points must be computed by identical manners. In contrast, if the assumption of conditional

independence given active points is extended to the testing case, FITC turns into another approach which is logically called Fully Independent Conditional (FIC) . FIC is equivalent to Gaussian Processes with the covariance function $k_{\text{FIC}}(\mathbf{x}_i, \mathbf{x}_j) = k_{\text{SOR}}(\mathbf{x}_i, \mathbf{x}_j) + \delta_{i,j}(k(\mathbf{x}_i, \mathbf{x}_j) - k_{\text{SOR}}(\mathbf{x}_i, \mathbf{x}_j))$, where $\delta_{i,j}$ is Kronecker delta function. The prior and predictive distribution implied by FIC is:

$$q_{\text{FIC}}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} & \mathbf{Q}_{\mathbf{X},\mathbf{X}_*}\\ \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}_*} + \mathbf{\Lambda}_*\end{bmatrix}\right) \tag{2.43}$$

$$\begin{aligned} q_{\text{FIC}}(\mathbf{f}_* \mid \mathbf{y}) = \mathcal{N}\big(\mathbf{f}_* \mid & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y},\\ & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}_*} + \mathbf{\Lambda}_* - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{Q}_{\mathbf{X},\mathbf{X}_*}\big). \end{aligned} \tag{2.44}$$

Partially Independent (Training) Conditional. Having compared the predictive distribution of DTC and FITC defined in equation 2.37 and 2.42, it is obvious that FITC is an improvement of DTC by remaining the exact diagonal elements of the covariance matrix. Relying on the unifying framework, Quiñonero Candela and Rasmussen (2005) have proposed a further improved approximation compared to FITC by extending the training conditional to have a block of diagonal covariance and remaining the exact testing covariance as defined in equation 2.26. Due to the usage of diagonal block covariance on training conditional, the approximation is called Partially Independent Training Conditionals (PITC) .

$$q_{\text{PITC}}(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{\mathbf{X},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{u}, \tilde{\mathbf{\Lambda}}\right) \tag{2.45}$$

$$q_{\text{PITC}}(\mathbf{f}_* \mid \mathbf{u}) = p(\mathbf{f}_* \mid \mathbf{u}) = \mathcal{N}\left(\mathbf{K}_{\mathbf{X}_*,\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}_*}\right). \tag{2.46}$$

, where $\tilde{\mathbf{\Lambda}} = bkdiag\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}\right]$ is a block diagonal matrix that is not clearly specified in Quiñonero Candela and Rasmussen (2005). An intuitive blocking structure is to group training points using clustering techniques as mentioned in Snelson and Ghahramani (2007). Similar to FITC, the approximation to joint prior of PITC is defined as:

$$q_{\text{PITC}}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \tilde{\mathbf{\Lambda}} & \mathbf{Q}_{\mathbf{X},\mathbf{X}_*}\\ \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*}\end{bmatrix}\right) \tag{2.47}$$

The approximation to the predictive distribution of PITC is identical to FITC defined in equation 2.42, except for the substitution of a block diagonal matrix.

$$\begin{aligned} q_{\text{PITC}}(\mathbf{f}_* \mid \mathbf{y}) = \mathcal{N}\big(\mathbf{f}_* \mid & \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \tilde{\mathbf{\Lambda}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y},\\ & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}\left(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \tilde{\mathbf{\Lambda}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{Q}_{\mathbf{X},\mathbf{X}_*}\big). \end{aligned} \tag{2.48}$$

As argued by Snelson and Ghahramani (2007), predictions obtained by PITC are empirically identical to FITC and FIC given a specified set of active positions and hyper-parameters. They have speculated that mean predictions of PITC are still just a weighted sum of basis functions centered on the same inducing inputs as in FITC or FIC, and the blocking structures on training covariance only changes the weights slightly. In addition, the structure of covariance of PITC defined in equation 2.47 means that the PITC approximation is not equivalent to a Gaussian Processes with a particular kernel function. To solve these problems, Snelson and Ghahramani (2007) relax the assumption of conditional independence between training and testing latent variables given inducing variable, i.e. do not approximate $p(\mathbf{f}, \mathbf{f}_* \mid \mathbf{u})$ by $q(\mathbf{f} \mid \mathbf{u})\, q(\mathbf{f}_* \mid \mathbf{u})$. They treat the training and testing inputs in the same manner, and put them into S blocks using clustering techniques. Consider a single testing input $\mathbf{x}_*$ which are assigned to block B_S, then the joint prior are approximate as:

$$p(\mathbf{f}, f_*) = \int p(\mathbf{f}, f_* \mid \mathbf{u})\, p(\mathbf{u})\, d\mathbf{u} \approx \int p(\mathbf{f}_{B_S}, f_* \mid \mathbf{u}) \prod_{s=1}^{S-1} p(\mathbf{f}_{B_s} \mid \mathbf{u})\, p(\mathbf{u})\, d\mathbf{u}. \tag{2.49}$$

According to the approximation, the assumption of using partial independence applies to both training and testing points. Therefore, this idea is logically called Partially Independence Conditional (PIC) . Thanks to the relaxation of conditional independence, PIC corresponds to a Gaussian Process with covariance matrix $\mathbf{K}_{\text{PIC}}$.

$$K_{\text{PIC}}(\mathbf{x}_i, \mathbf{x}_j) = Q(\mathbf{x}_i, \mathbf{x}_j) + \psi(\mathbf{x}_i, \mathbf{x}_j)\left[K(\mathbf{x}_i, \mathbf{x}_j) - Q(\mathbf{x}_i, \mathbf{x}_j)\right]. \tag{2.50}$$

, where

$$\psi(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 \text{ if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{are in the same block} \\ 0 \text{ otherwise.} \end{cases} \tag{2.51}$$

The predictive distribution implied by PIC is identical to the exact predictive distribution, except for the alternation of $\mathbf{K}$ by $\mathbf{K}_{\text{PIC}}$.

2.3.2 Posterior Approximations

Weakness of Prior Approximations. As alluded to previously, the aforementioned algorithms complying with the idea of prior approximation operate as an exact Gaussian Processes with a particular kernel function or an approximation to covariance matrices, i.e. SOR (Smola and Bartlett, 2001) and FIC (Snelson and Ghahramani, 2006) and PIC (Snelson and Ghahramani, 2007). Suppose we would like to employ M inducing variables which are latent values at some auxiliary inputs $\mathbf{Z}$ to approximate the GP prior.

The quality of these sparse approximations depends on the optimization of $\mathbf{Z}$ and hyper-parameters, i.e. kernel's parameters and variance noise (for Gaussian likelihood). An approximation to the true marginal likelihood defined in 2.1 allows us to select $\mathbf{Z}$ and other hyper-parameter using a gradient-based iterative method. For example, consider a zero-mean GP, Projected Process approximation (PP) (Seeger et al., 2003) and Sparse Gaussian Processes using Pseudo-points (SGPP) (Snelson and Ghahramani, 2006) following the idea of prior approximation replace the logarithm of GP marginal likelihood $F_{\text{GP}} = \log\left[p\left(\mathbf{y} \mid \mathbf{X}\right)\right]$ by F_{PP} and F_{SGPP}:

$$\text{GP: } F_{\text{GP}} = \log\left[\mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right)\right]. \tag{2.52}$$

$$\text{PP}: F_{\text{PP}} = \log\left[\mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{Q}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right)\right]. \tag{2.53}$$

$$\text{SGPP}: F_{\text{SGPP}} = \log\left[\mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} + \sigma_n^2\mathbf{I}\right)\right]. \tag{2.54}$$

where we recall that $\mathbf{Q}_{\mathbf{X},\mathbf{X}} = \mathbf{K}_{\mathbf{X},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{X}}$ is the Nyström approximation to $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ using inducing inputs $\mathbf{Z}$. $\mathbf{\Lambda} = \text{Diag}\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}\right]$ is the difference on diagonal elements between the true kernel values and approximated ones. Observe F_{PP} and F_{SGPP}, the covariance of approximate marginal likelihood are parameterized by inducing inputs $\mathbf{Z}$. While it is tempting to think that the introduction of $\mathbf{Z}$ in kernel function improves the representational power of approximate GPs, the highly-parameterized form probably lead to an over-fitting problem because the continuous optimization of F_{PP} and F_{SGPP} with respect to $\mathbf{Z}$ does not reliably approximate true GP.

Main idea of Posterior Approximations. In order to deal with the lack of consideration of the convergence between true GP and approximate ones, we intuitively find $\mathbf{Z}$ by minimizing the *distance* of the approximated predictive distributions produced by the inducing points and the true ones. Further, the idea also allows us to access the divergence between the true GP and sparse approximation to GP. Since both of predictive distribution $p\left(\mathbf{f}_* \mid \mathbf{y}\right)$ and posterior $p\left(\mathbf{f} \mid \mathbf{y}\right)$ are conditional prior given observations, the selection of $\mathbf{Z}$ based upon the idea is equivalent to minimize the Kullback-Leibler divergence between the approximate posterior $q\left(\mathbf{f}\right)$ and the true posterior $p\left(\mathbf{f} \mid \mathbf{y}\right)$.

Starting from the true GP conditional prior over arbitrary auxiliary variable $\mathbf{v}$ given observations $\mathbf{y}$, we construct the approximate GP posterior using M inducing points. We can express the conditional prior $p\left(\mathbf{v} \mid \mathbf{y}\right)$ by integrating out inducing variables $\mathbf{u}$ and training latent values $\mathbf{f}$ as follows:

$$p\left(\mathbf{v} \mid \mathbf{y}\right) = \int p\left(\mathbf{v} \mid \mathbf{u}, \mathbf{f}\right) p\left(\mathbf{f} \mid \mathbf{u}, \mathbf{y}\right) p\left(\mathbf{u} \mid \mathbf{y}\right) d\mathbf{f} d\mathbf{u}. \tag{2.55}$$

By assuming $\mathbf{u}$ completely capture the relation between $\mathbf{v}$ and $\mathbf{f}$, it holds that $p(\mathbf{v} \mid \mathbf{u}, \mathbf{y}) = p(\mathbf{v} \mid \mathbf{u})$. Thanks to the assumption of conditional independence, the variable $\mathbf{f}$ only appears in $p(\mathbf{f} \mid \mathbf{u}, \mathbf{y})$, and therefore, $\mathbf{f}$ is canceled out as $\int p(\mathbf{f} \mid \mathbf{u}, \mathbf{y})\, d\mathbf{f} = 1$. Subsequently, the above $p(\mathbf{v} \mid \mathbf{y})$ can be written as $q(\mathbf{v})$:

$$q(\mathbf{v}) = \int p(\mathbf{v} \mid \mathbf{u})\, q(\mathbf{u})\, d\mathbf{u} = \int q(\mathbf{v}, \mathbf{u})\, d\mathbf{u}. \tag{2.56}$$

where $q(\mathbf{v}) = p(\mathbf{v} \mid \mathbf{y})$ and $q(\mathbf{u}) = p(\mathbf{u} \mid \mathbf{y})$. Practically speaking, it is infeasible to find inducing variables $\mathbf{u}$ which are sufficient statistics for the parameters $\mathbf{f}$. Thus, $q(\mathbf{v})$ should be expected as an approximation to $p(\mathbf{v} \mid \mathbf{y})$. Subsequently, the $q(\mathbf{u})$ can be represented by a parameterized form.

Since the joint variable $[\mathbf{z}, \mathbf{y}]^T$ and $[\mathbf{z}, \mathbf{u}]^T$ follow a GP, the conditional priors of $p(\mathbf{z} \mid \mathbf{y})$ and $p(\mathbf{z} \mid \mathbf{u})$ are also Gaussian densities. Intuitively, $q(\mathbf{z})$ which is the approximation to $p(\mathbf{z} \mid \mathbf{y})$ should be also a Gaussian. Thanks to the equation 2.56 and the conjugacy properties, we see that assuming $q(\mathbf{u})$ a variational Gaussian distribution defined by a mean vector $\mathbf{m}$ and covariance matrix $\mathbf{S}$ turns $q(\mathbf{v})$ to be a Gaussian. Introducing $\tilde{\mathbf{X}}$ as the indices of $\mathbf{v}$, we can express $q(\mathbf{v})$ under a closed form:

$$q(\mathbf{v}) = \mathcal{N}\left(\mathbf{A}\mathbf{m}, \mathbf{K}_{\tilde{\mathbf{X}},\tilde{\mathbf{X}}} + \mathbf{A}\left(\mathbf{S} - \mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)\mathbf{A}\right). \tag{2.57}$$

where $\mathbf{A} = \mathbf{K}_{\tilde{\mathbf{X}},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}$. Since $\mathbf{v}$ is an arbitrary variable representing all latent function values, $q(\mathbf{v})$ can be perceived as an approximation to GP posterior $q(\mathbf{f})$ or predictive distributions $q(\mathbf{f}_*)$.

Turning to the idea of posterior approximation, all parameters $\boldsymbol{\theta}$, e.g. inducing inputs or hyper-parameters, are selected to minimize the Kullback-Leibler divergence between the approximate posterior $q(\mathbf{f})$ and the true posterior $p(\mathbf{f} \mid \mathbf{y})$. Equivalently, we can minimize a distance between the augmented variational posterior $q(\mathbf{f}, \mathbf{u})$ defined in equation 2.56, i.e. $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u})\, \phi(\mathbf{u})$ and the augmented true posterior $p(\mathbf{f}, \mathbf{u} \mid \mathbf{y})$.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathrm{KL}\left[q(\mathbf{f}, \mathbf{u}) \mid\mid p(\mathbf{f}, \mathbf{u} \mid \mathbf{y})\right] \tag{2.58}$$

Taking further analysis, we see that $\mathrm{KL}\left[q(\mathbf{f}, \mathbf{u}) \mid\mid p(\mathbf{f}, \mathbf{u} \mid \mathbf{y})\right]$ can be represented as:

$$\mathrm{KL}\left[q(\mathbf{f}, \mathbf{u}) \mid\mid p(\mathbf{f}, \mathbf{u} \mid \mathbf{y})\right] = \log\left[p(\mathbf{y})\right] - \mathrm{E}_{q(\mathbf{f},\mathbf{u})} \log\left[\frac{p(\mathbf{f}, \mathbf{u}, \mathbf{y})}{q(\mathbf{f}, \mathbf{u})}\right]. \tag{2.59}$$

Since $\log[p(\mathbf{y})]$ is constant for $q(\mathbf{f}, \mathbf{u})$, learning all parameters can be inferred by maximizing F_q defined as follows:

$$F_q = \mathrm{E}_{q(\mathbf{f},\mathbf{u})} \log\left[\frac{p(\mathbf{f},\mathbf{u},\mathbf{y})}{q(\mathbf{f},\mathbf{u})}\right] = \int p(\mathbf{f} \mid \mathbf{u})\, q(\mathbf{u}) \log\left[\frac{p(\mathbf{y} \mid \mathbf{f})\, p(\mathbf{u})}{q(\mathbf{u})}\right] d\mathbf{f} d\mathbf{u}. \tag{2.60}$$

Sparse Variational Gaussian Processes Regression. The most well-known representative following the idea of posterior approximation is proposed by Titsias (2009) using variational inference technique (Blei et al., 2017). In the approach, the lower bound to marginal likelihood is developed from F_q defined in 2.60. To derive a tighter bound, they firstly maximize the bound F_q by analytically solving for the optimal choice of the variational distribution $q^*(\mathbf{u})$. By differentiating 2.60 with respect to $q(\mathbf{u})$ and using Gaussian likelihood $p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma_n^2 \mathbf{I})$, the optimal $q^*(\mathbf{u})$ is derived as follows:

$$q^*(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m}^*, \mathbf{S}^*), \text{ where,} \tag{2.61}$$

$$\mathbf{m}^* = \sigma_n^2 \mathbf{K}_{\mathbf{Z},\mathbf{Z}} \left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}} + \sigma_n^2 \mathbf{K}_{\mathbf{Z},\mathbf{X}} \mathbf{K}_{\mathbf{X},\mathbf{Z}}\right)^{-1} \mathbf{K}_{\mathbf{Z},\mathbf{X}} \mathbf{y} \tag{2.62}$$

$$\mathbf{S}^* = \mathbf{K}_{\mathbf{Z},\mathbf{Z}} \left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}} + \sigma_n^2 \mathbf{K}_{\mathbf{Z},\mathbf{X}} \mathbf{K}_{\mathbf{X},\mathbf{Z}}\right)^{-1} \mathbf{K}_{\mathbf{Z},\mathbf{Z}} \tag{2.63}$$

By replacing $q^*(\mathbf{u})$ into the bound F_q defined in Equation 2.60, we obtain the lower bound of Sparse Gaussian Processes for Regression (SGPR) proposed by Titsias (2009):

$$F_{\text{SGPR}} = \log\left[\mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \sigma_n^2 \mathbf{I} + \mathbf{Q}_{\mathbf{X},\mathbf{X}}\right)\right] - \frac{1}{2\sigma_n^2} \mathrm{Tr}(\mathbf{\Lambda}). \tag{2.64}$$

where we recall that $\mathbf{Q}_{\mathbf{X},\mathbf{X}} = \mathbf{K}_{\mathbf{X},\mathbf{Z}} \mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1} \mathbf{K}_{\mathbf{Z},\mathbf{X}}$ is the Nyström approximation to $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ using inducing inputs $\mathbf{Z}$, and $\mathbf{\Lambda} = \mathrm{Diag}[\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}]$ is the difference on diagonal elements between the true kernel values and approximated ones. Observe the approximation to GP marginal likelihood of the approach of Projected Process Approximation (PP) or Deterministic Training Conditions (DTC) defined in 2.53, we can rewrite F_{SGPR} in terms of F_{PP}:

$$F_{\text{SGPR}} = F_{\text{PP}} - \frac{1}{2\sigma_n^2} Tr\left(\mathrm{Diag}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}\right)\right). \tag{2.65}$$

It is obvious that SGPR differs DTC only by trace term, which have a significant impact on the inference. Intuitively, the $Tr\left(\mathrm{Diag}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}\right)\right)$ represents the total variance of predicting the latent variables $\mathbf{f}$ given $\mathbf{u}$. When maximizing the bound F_{SGPR}, the positive trace term should be decreased, and, in particular, the fact of the trace is zero means that $\mathbf{u}$ recover the full GP. Therefore, the trace term not only seeks to deliver a good inducing set but

also prevents SGPR from overfitting.

Stochastic Variational Inference for Gaussian Processes. A downside of SGPR proposed by Titsias (2009) is that the computational and storage cost depends on the training size N linearly. On the inspection of the bound F_{SGPR} defined in equation 2.64, each training iteration of SGPR requires the budget of $\mathcal{O}(NM^2)$ for computation and $\mathcal{O}(NM)$. These costs come from the linear algebraic operation appearing the computation of $\mathbf{Q}_{\mathbf{X},\mathbf{X}}$, i.e. the matrix inversion of $\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}$ and the matrix multiplication $\mathbf{K}_{\mathbf{X},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{X}}$. Though the reduction of SGPR on computation and memory requirement are impressive, these demands are quickly prohibitive for big data, where the training size N reaches to many millions or billions.

In order to overcome the dependency of complexities on training size, Hensman et al. (2013) have employed Stochastic Variational Inference on Gaussian Processes. This approach is, therefore, abbreviated by SVI . While Titsias' bound are derived by replacing $q(\mathbf{u})$ by optimal distribution for inducing variable $q^*(\mathbf{u})$ defined in equation 2.61, SVI (Hensman et al., 2013) parameterize the variational distribution $q(\mathbf{u})$ by a Gaussian density $\mathcal{N}(\mathbf{m}, \mathbf{S})$. Substituting $\mathcal{N}(\mathbf{m}, \mathbf{S})$ for $q(\mathbf{u})$ in the general bound F_q defined in the equation 2.60, the bound F_{SVI} are obtained as follows:

$$F_{\text{SVI}} = \sum_{i=1}^{N} \left\{ \log \mathcal{N}\left(y_n \mid \mathbf{K}_{\mathbf{x}_i,\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{m}, \sigma_n^2\right) - \frac{1}{2\sigma_n^2}\mathbf{K}_{\mathbf{x}_i,\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{S}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{x}_i} - \frac{1}{2\sigma_n^2}\left(\mathbf{K}_{\mathbf{x}_i,\mathbf{x}_i} - Q_{\mathbf{x}_i,\mathbf{x}_i}\right) \right\} - \text{KL}\left(q(\mathbf{u}) \,||\, p(\mathbf{u})\right). \tag{2.66}$$

Due to the Gaussian form of $q(\mathbf{u})$, the KL term can be expressed analytically with the computational cost of $\mathcal{O}(M^3)$. The most important property of F_{SVI} is that it can be written as a sum of N terms, each of them corresponds to one pair of input and output $(\mathbf{x}_i, y_i)$. This allows us to perform stochastic gradient ascent by using a mini-batch $\mathcal{I}$ as follows:

$$F_{\text{SVI}} \approx \frac{N}{|\mathcal{I}|} \sum_{(\mathbf{x}_i,y_i)\in\mathcal{I}} \left\{ \log \mathcal{N}\left(y_i \mid \mathbf{K}_{\mathbf{x}_i,\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{m}, \sigma_n^2\right) - \frac{1}{2\sigma_n^2}\mathbf{K}_{\mathbf{x}_i,\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{S}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{x}_i} - \frac{1}{2\sigma_n^2}\left(\mathbf{K}_{\mathbf{x}_i,\mathbf{x}_i} - Q_{\mathbf{x}_i,\mathbf{x}_i}\right) \right\} - \text{KL}\left(q(\mathbf{u}) \,||\, p(\mathbf{u})\right). \tag{2.67}$$

Apart from accelerating the computation cost by applying stochastic variational inference, the factorization over training examples allows the combina-

tion of SVI and non-Gaussian likelihood. As a consequence, a more general approach of SVI has also proposed by Hensman et al. (2015a), which is called Scalable Variational Gaussian Processes (SVGP) . The bound F_{SVGP} can be derived easily by rewriting F_q in equation 2.60:

$$\begin{aligned} F_{\text{SVGP}} = F_q &= \int p\left(\mathbf{f} \mid \mathbf{u}\right) q\left(\mathbf{u}\right) \log \left[\frac{p\left(\mathbf{y} \mid \mathbf{f}\right) p\left(\mathbf{u}\right)}{q\left(\mathbf{u}\right)}\right] d\mathbf{f} d\mathbf{u} \\ &= \mathrm{E}_{q(\mathbf{f})} \log p\left(\mathbf{y} \mid \mathbf{f}\right) - \mathrm{KL}\left(q\left(\mathbf{u}\right) \,||\, p\left(\mathbf{u}\right)\right) \\ &= \sum_{i=1}^{N} \mathrm{E}_{q(f_i)} \log p\left(y_i \mid f_i\right) - \mathrm{KL}\left(q\left(\mathbf{u}\right) \,||\, p\left(\mathbf{u}\right)\right) \\ &\approx \frac{N}{|\mathcal{I}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{I}} \mathrm{E}_{q(f_i)} \log p\left(y_i \mid f_i\right) - \mathrm{KL}\left(q\left(\mathbf{u}\right) \,||\, p\left(\mathbf{u}\right)\right). \end{aligned} \tag{2.68}$$

, where $q\left(f_i\right)$ is calculated as $q\left(\mathbf{v}\right)$ defined in equation 2.57. In case the likelihood $p\left(y_i \mid f_i\right)$ is Gaussian, the variational expectation term can be expressed analytically. In general, the one-dimensional integrals of the log-likelihood can be computed by Gauss-Hermite quadrature as in Hensman et al. (2015a).

Further Improvement. The approaches of posterior approximation (Titsias, 2009; Hensman et al., 2013, 2015a) can be further improved in various ways. The first direction is to apply a Bayesian treatment to all kernel hyperparameters rather than optimizing them, which is prone to overfitting (Titsias and Lazaro-Gredilla, 2013; Hensman et al., 2015b; Yu et al., 2017). Another extension is to allow to work with a non-Gaussian posterior, e.g. mixture of Gaussians (Nguyen and Bonilla, 2014a), or a free-form posterior (Hensman et al., 2015b).

2.3.3 Structure Exploiting Approximations

Main idea. Consider a GP with Gaussian likelihood $p\left(y_i \mid f_i\right) = \mathcal{N}\left(y_i \mid f_i, \sigma_n^2\right)$, the gradients of logarithm of the marginal likelihood $p\left(\mathbf{y} \mid \mathbf{X}\right)$ with respect to an arbitrary trainable parameter ψ is as follows:

$$\frac{\partial \log\left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}\right)\right]}{\psi} = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{K}_{\sigma_n^2}^{-1} \frac{\partial \mathbf{K}_{\sigma_n^2}}{\partial \psi}\right) + \frac{1}{2}\mathbf{y}^T \mathbf{K}_{\sigma_n^2}^{-1} \frac{\partial \mathbf{K}_{\sigma_n^2}}{\partial \psi} \mathbf{K}_{\sigma_n^2}^{-1} \mathbf{y}. \tag{2.69}$$

where $\mathbf{K}_{\sigma_n^2} = \mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}$. Traditionally, the Cholesky decomposition is applied to factorize $\mathbf{K}_{\sigma_n^2} = \mathbf{L}\mathbf{L}^T$ which cost $\mathcal{O}\left(N^3\right)$ (Golub and Van Loan, 1996). Therefore, the computational problems of these gradients start to arise when N exceed a few thousands. It is possible to enhance the scalability of the computations by imposing a special algebraic structure on the kernel matrix

$\mathbf{K}_{\mathbf{X},\mathbf{X}}$. A well-known approach following the idea of structural exploitation is to use Kronecker product with the assumption of grid-structure dataset and tensor product kernel (Saatçi, 2011; Gilboa et al., 2015).

Grid-structured data. According to the exposition of Chapter 5 in Saatçi's dissertation, we assume all input points $\mathbf{X}$ are located on a Cartesian grid, i.e.

$$\mathbf{X} = \mathbf{X}_1 \times \cdots \times \mathbf{X}_D \tag{2.70}$$

, where $\mathbf{X}_d$ represents all distinct input locations along dimension d, and operator $\times$ indicates the Cartesian product between vectors. The number of elements of the vector $\mathbf{X}_d$ is generally arbitrary, i.e. we can say that $\mathbf{X}_d \in \mathbb{R}^{G_d}$ where G_d is the size of vector $\mathbf{X}_d$. The definition of Cartesian product entails that $\mathbf{X}$ is restricted to contain exactly $\prod_{d=1}^{D} G_d$ points which are put on the D-dimensional Cartesian grid. Though a grid-structured data can enable the computational acceleration, the number of data points grows exponentially with dimensions, and, consequently, the limitation of the computational resource is quickly reached. Therefore, the speed-up of GP using the idea of grid-structured data is feasible with few dimensions. For example, following Saatçi (2011), the applicability of GP on multidimensional grid data is restricted with the dimension which is less than 8. Despite severely suffering from the curse of dimensionality, this structure arises naturally in several spatial-temporal problems such as climate modeling, where the input points generally denote latitude and longitude coordinates that can be further augmented with some periodically spaced time dimension. Multimedia such as images and videos are also likely to inherently have such structure.

Tensor product kernel. In this section, the covariance functions are assumed to be tensor product kernels, which compute the covariance as a separable product over dimensions. Introducing two D-dimensional covariates $\mathbf{x}_i$ and $\mathbf{x}_j$ belonging to the grid-structure input space $\mathbf{X}$ mentioned above, the covariance between $\mathbf{x}_i$ and $\mathbf{x}_j$ can be written as:

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \prod_{d=1}^{D} k_d\left(\mathbf{x}_{i,d}, \mathbf{x}_{j,d}\right). \tag{2.71}$$

where $\mathbf{x}_{i,d} \in \mathbf{X}_d$ is the d-th element of input $\mathbf{x}_i$ and $k_d\left(.,.\right)$ is any symmetric positive definite function which is described in section 2.2.2.

Algebraic advantages of the Kronecker method. Introducing $\mathbf{A}$ as $m \times n$ matrix and $\mathbf{B}$ as $p \times q$ matrix, the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$,

denoted by $\mathbf{A} \otimes \mathbf{B}$, is an $mp \times nq$ matrix having the following form:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m,1}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{bmatrix} \tag{2.72}$$

, more explicitly:

$$\mathbf{A}\otimes\mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \dots & a_{11}b_{1q} & \dots & \dots & a_{1n}b_{11} & a_{1n}b_{12} & \dots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \dots & a_{11}b_{2q} & \dots & \dots & a_{1n}b_{21} & a_{1n}b_{22} & \dots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \dots & a_{11}b_{pq} & \dots & \dots & a_{1n}b_{p1} & a_{1n}b_{p2} & \dots & a_{1n}b_{pq} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \dots & a_{m1}b_{1q} & \dots & \dots & a_{mn}b_{11} & a_{mn}b_{12} & \dots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \dots & a_{m1}b_{2q} & \dots & \dots & a_{mn}b_{21} & a_{mn}b_{22} & \dots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \dots & a_{m1}b_{pq} & \dots & \dots & a_{mn}b_{p1} & a_{mn}b_{p2} & \dots & a_{mn}b_{pq} \end{bmatrix} \tag{2.73}$$

For the sake of clarity, we mention the basic properties of Kronecker product with square matrices, which is helpful for a forthcoming explanation.

$$\text{Bilinearity: } \mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C} \tag{2.74}$$

$$\text{Associativity: } (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) \tag{2.75}$$

$$\text{Mixed-product property: } (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD} \tag{2.76}$$

$$\text{Inverse: } (\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1} \tag{2.77}$$

$$\text{Transpose: } (\mathbf{A} \otimes \mathbf{B})^{T} = \mathbf{A}^{T} \otimes \mathbf{B}^{T} \tag{2.78}$$

$$\text{Trace: } \operatorname{Tr}(\mathbf{A} \otimes \mathbf{B}) = \operatorname{Tr}(\mathbf{A}) \operatorname{Tr}(\mathbf{B}) \tag{2.79}$$

$$\text{Determinant: } \det(\mathbf{A} \otimes \mathbf{B}) = (\det \mathbf{A})^{D_A} (\det \mathbf{B})^{D_B} \tag{2.80}$$

$$\text{Vec: } Vec\left(\mathbf{CXB}^{T}\right) = (\mathbf{B} \otimes \mathbf{C})\, Vec(\mathbf{X}). \tag{2.81}$$

, where D_A and D_B are dimensions of matrices $\mathbf{A}$ and $\mathbf{B}$. Introducing $\mathbf{X}$ as m-by-n matrix, $Vec(\mathbf{X})$ denotes flatten operator yielding mn-dimensional vector.

Thanks to the assumptions of grid-structured data and tensor product kernel, the full covariance matrix for points on the grid can be evaluated by Kronecker product of a sequence of kernels:

$$\mathbf{K}_{\mathbf{X},\mathbf{X}} = \mathbf{K}_1(\mathbf{X}_1, \mathbf{X}_1) \otimes \mathbf{K}_2(\mathbf{X}_2, \mathbf{X}_2) \otimes \cdots \otimes \mathbf{K}_D(\mathbf{X}_D, \mathbf{X}_D) \tag{2.82}$$

, where $\mathbf{K}_d$ is $G_d \times G_d$ covariance matrix of the vector of scalar input locations $\mathbf{X}_d$. In order to see how Kronecker product gain the benefit in GP computations, we remind the logarithm of the marginal likelihood of a zero-mean GP with the Gaussian likelihood $p(y_i \mid f_i) = \mathcal{N}(y_i \mid f_i, \sigma_n^2)$:

$$\log p(\mathbf{y} \mid \mathbf{X}) = -\frac{1}{2}\log\left|\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right| - \frac{1}{2}\mathbf{y}^T\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} - \frac{N}{2}\log 2\pi. \tag{2.83}$$

It is infeasible to access the logarithm marginal likelihood of GP regression on $\mathbf{X}$ containing $N = \prod_{d=1}^{D} G_d$ points due to the computational bottlenecks from the algebraic operations, i.e. the inversion and matrix-vector multiplication $\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}$ and logarithm of determinant $\log|\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}|$. The original computational and storage cost are $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ respectively. Due to the nice properties of Kronecker product, the complexity of learning and inference turns out $\mathcal{O}\left(DN^{1+\frac{1}{D}}\right)$ and $\mathcal{O}\left(DN^{\frac{2}{D}}\right)$ for storage. In the next section, I will analyze and explain why Kronecker product can lead to the improvements.

These reductions come from the fast computation of eigendecomposition $\mathbf{K}_{\mathbf{X},\mathbf{X}} = \mathbf{Q}\mathbf{V}\mathbf{Q}^T$, where $\mathbf{V}$ is the diagonal matrix constructed by corresponding eigenvalues v_i, and $\mathbf{Q}$ is the square matrix whose i-th column is the eigenvector q_i of $\mathbf{K}_{\mathbf{X},\mathbf{X}}$, and therefore, $\mathbf{Q}$ is guaranteed to be an orthogonal matrix, and consequently, $\mathbf{Q}^{-1} = \mathbf{Q}^T$. Since $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ can be expressed by Kronecker product, the computation for matrices $\mathbf{Q}$ and $\mathbf{V}$ is accelerated by separately carrying out the eigendecomposition of $\mathbf{K}_1(\mathbf{X}_1, \mathbf{X}_1), \ldots, \mathbf{K}_D(\mathbf{X}_D, \mathbf{X}_D)$. Denoting $\mathbf{Q}_d$ as matrix containing eigenvectors of $\mathbf{K}_d(\mathbf{X}_d, \mathbf{X}_d)$ and $\mathbf{V}_d$ as a diagonal matrix of eigenvalues of $\mathbf{K}_d(\mathbf{X}_d, \mathbf{X}_d)$, i.e. $\mathbf{K}_d(\mathbf{X}_d, \mathbf{X}_d) = \mathbf{Q}_d\mathbf{V}_d\mathbf{Q}_d^T$, matrix $\mathbf{Q}$ and $\mathbf{V}$ can be expressed as Kronecker products by using the Mixed-product property defined at 2.76. Actually, $\mathbf{V}_d$ and $\mathbf{V}$ are diagonal matrices, and $\mathbf{V}$ are constructed by concatenating diagonal elements of $\mathbf{V}_d$.

$$\mathbf{Q} = \mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_D \text{ , and } \mathbf{V} = \mathrm{Diag}\left(\mathrm{Diag}(\mathbf{V}_1)^T, \ldots, \mathrm{Diag}(\mathbf{V}_D)^T\right) \tag{2.84}$$

Due to the Vec property mentioned at 2.81, the fast matrix vector multiplication are enabled by using the Algorithm kron-mvn mentioned in Saatçi's dissertation (Saatçi, 2011).

In order to analyze the complexity of kron_mvm conveniently, I assume all $\mathbf{A}_d$ have the same dimensions. Similarly, the algorithm kron_mvm also works with matrices $\{\mathbf{A}\}_{d=1}^{D}$ with various sizes, i.e. $\mathbf{A}_d \in \mathbb{R}^{G_d \times G_d}$. Consider the iterative steps appear in the loop, the computational cost mainly relies on the matrix multiplication $\mathbf{A}_d\mathbf{X}$ which requires $\mathcal{O}(NG)$ or $\mathcal{O}\left(N^{1+\frac{1}{D}}\right)$. The loop

Algorithm 1 Fast Matrix Vector Multiplication with Kronecker Product - kron_mvm.
Input: G-by-G matrices $\mathbf{A}_1, \ldots, \mathbf{A}_D$, N-dimensional vector $\mathbf{b}$ where $N = G^D$
Output: $\boldsymbol{\alpha} = \left(\otimes_{d=1}^{D} \mathbf{A}_d\right) \mathbf{b}$.

1: $\boldsymbol{\alpha} \leftarrow \mathbf{b}$.
2: **for** $d \leftarrow D$ to 1 **do**
3: $\quad \mathbf{X} \leftarrow$ reshape $(\boldsymbol{\alpha}, G, N/G)$;
4: $\quad \mathbf{Z} \leftarrow \mathbf{A}_d \mathbf{X}$;
5: $\quad \boldsymbol{\alpha} \leftarrow vec\left(\mathbf{Z}^T\right)$
6: **end for**

is repeated D times, therefore, the ultimate budget is $\mathcal{O}\left(DN^{1+\frac{1}{D}}\right)$.

Turning to the matrix-vector multiplication $\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y}$ appearing in the logarithm of marginal likelihood, it can be rewritten in terms of $\mathbf{Q}$ and $\mathbf{V}$. Thanks to the property of *Transpose* defined at 2.78, $\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y}$ can be further represented by Kronecker product:

$$\begin{aligned} \left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y} &= \mathbf{Q} \left(\mathbf{V} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{Q}^T \mathbf{y} \\ &= \left(\mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_D\right) \left(\mathbf{V} + \sigma_n^2 \mathbf{I}\right)^{-1} \left(\mathbf{Q}_1^T \otimes \cdots \otimes \mathbf{Q}_D^T\right) \mathbf{y} \end{aligned} \tag{2.85}$$

With the above expression, the matrix vector multiplication can be solved efficiently using the following steps:

$$\begin{aligned} &\boldsymbol{\alpha} \leftarrow \text{kron\_mvm}\left(\left[\mathbf{Q}_1^T, \ldots, \mathbf{Q}_D^T\right], \mathbf{y}\right) \\ &\boldsymbol{\alpha} \leftarrow \left(\mathbf{V} + \sigma_n^2 \mathbf{I}\right)^{-1} \boldsymbol{\alpha} \\ &\boldsymbol{\alpha} \leftarrow \text{kron\_mvm}\left(\left[\mathbf{Q}_1, \ldots, \mathbf{Q}_D\right], \boldsymbol{\alpha}\right) \end{aligned} \tag{2.86}$$

where kron_mvm is a procedure detailed in Algorithm 1. Remind that $\mathbf{V}$ is diagonal matrix containing the eigenvalues of block covariances $\left\{\mathbf{K}_d\left(\mathbf{X}_d, \mathbf{X}_d\right)\right\}_{d=1}^{D}$, the matrix $\mathbf{V} + \sigma_n^2 \mathbf{I}$ is also diagonal, and, therefore, its inversion can be computed with linear complexity.

The fast eigendecomposition of $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ also speeds up the computation of logarithm of determinant of $\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}$. Denoting $v_1, \ldots, v_N$ the diagonal elements of $\mathbf{V}$, we know that $\{v_i\}_{i=1}^{N}$ are eigenvalues of $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ as $\mathbf{K}_{\mathbf{X},\mathbf{X}} = \mathbf{Q}\mathbf{V}\mathbf{Q}^T$. Due to the definition of eigenvalue, there is a relation between matrix $\mathbf{K}_{\mathbf{X},\mathbf{X}}$, its eigenvalue v_i and its corresponding eigenvector $\mathbf{q}_i$: $\mathbf{K}_{\mathbf{X},\mathbf{X}} \mathbf{q}_i = v_i \mathbf{q}_i$, then $\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right) \mathbf{q}_i = v_i \mathbf{q}_i + \sigma_n^2 \mathbf{I} \mathbf{q}_i = \left(v_i + \sigma_n^2\right) \mathbf{q}_i$. Therefore, it can be derived that if v_i is an eigenvalue of $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ then $v_i + \sigma_n^2$ is also an eigenvalue

of $\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}$. In consequence, the logarithm of determinant of $\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}$ is reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}\left(DN^{\frac{3}{D}}\right)$ which are the cost for eigendecomposition of D matrices $\{\mathbf{K}_d(\mathbf{X}_d, \mathbf{X}_d)\}_{d=1}^{D}$.

$$\log\left|\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right| = \sum_i \log\left(v_i + \sigma_n^2\right). \tag{2.87}$$

Structural Kernel Interpolation. Despite their impressive computational acceleration of Kronecker-based methodology presented above, the main limitation of this approach is the restriction of grid-structured data. However, most datasets will not satisfy this requirement, making the application of such techniques narrow. In order to relax the condition of having observations at all possible input locations in the grid, there are several attempts such that missing observations and incomplete grid are also permitted (Flaxman et al., 2015; Wilson et al., 2014). Ultimately, Wilson and Nickisch (2015) have extended the concept of the Kronecker method to a general scenario with the proposal of Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP) . This method constrains that the set of inducing positions $\mathbf{Z}$ constructs a complete multidimensional grid. Consider D-dimensional problems and introduce $\mathbf{Z}_d$ as a vector containing distinct inducing locations along with dimension d, we again define $\mathbf{Z}$ as Cartesian product of $\mathbf{Z}_1, \ldots, \mathbf{Z}_D$, i.e. $\mathbf{Z} = \mathbf{Z}_1 \otimes \cdots \otimes \mathbf{Z}_D$.

Similarly, the size of $\mathbf{Z}$ is $M = \prod_{d=1}^{D} G_d$ where G_d is the number of elements in the vector $\mathbf{Z}_d$. By utilizing the tensor product kernel defined above, the Kronecker idea enables the fast algebraic operations of the covariance matrix of inducing points. Nevertheless, setting a massive M could be problematic due to the time-consuming operations associated with the cross-covariance $\mathbf{K}_{\mathbf{X},\mathbf{Z}}$ between design matrix $\mathbf{X}$ and inducing locations $\mathbf{Z}$. For example, the full covariance matrix $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ which can be expressed by Nyström approximation $\mathbf{K}_{\mathbf{X},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{X}}$ dominate the computations with quadratic complexity to M, i.e. $\mathcal{O}(NM^2)$. Instead of computing directly $\mathbf{K}_{\mathbf{X},\mathbf{Z}}$, it is estimated by interpolating on the $M \times M$ covariance matrix $\mathbf{K}_{\mathbf{Z},\mathbf{Z}}$. For example, if we would like to estimate $k(\mathbf{x}, \mathbf{z}_j)$, for point $\mathbf{x}$ and inducing input $\mathbf{z}_j$, we can start by finding the two inducing points $\mathbf{z}_a$ and $\mathbf{z}_b$ which are the two closest to $\mathbf{x}$. Then, we can estimate $k(\mathbf{x}, \mathbf{z}_j)$ by $\tilde{k}(\mathbf{x}, \mathbf{z}_j) = wk(\mathbf{z}_j, \mathbf{z}_a) + (1-w)k(\mathbf{z}_j, \mathbf{z}_b)$, where w and $1-w$ are represented the relative distance from $\mathbf{x}$ to $\mathbf{z}_a$ and $\mathbf{z}_b$. Generally, the cross covariance $\mathbf{K}_{\mathbf{X},\mathbf{Z}}$ between design matrix $\mathbf{X}$ and inducing points $\mathbf{Z}$ can be interpolated by:

$$\mathbf{K}_{\mathbf{X},\mathbf{Z}} \approx \tilde{\mathbf{K}}_{\mathbf{X},\mathbf{Z}} = \mathbf{W}\mathbf{K}_{\mathbf{Z},\mathbf{Z}} \tag{2.88}$$

While M is expected very large in the scenario, $\mathbf{W}$ is constraint to be extremely sparse. There are several options to construct matrix $\mathbf{W}$ based upon various strategy including (i) local linear interpolation where each row of $\mathbf{W}$ contains only 2 non-zero entries or (ii) local cubic interpolation for greater accuracy with 4 non-zero elements per row.

As a consequence, from the Nyström approximation to full covariance $\mathbf{K}_{\mathbf{X},\mathbf{X}}$, a further estimation can be obtained by substituting $\tilde{\mathbf{K}}_{\mathbf{X},\mathbf{Z}}$ for $\mathbf{K}_{\mathbf{X},\mathbf{Z}}$. This general approach of approximation is so called Structured Kernel Interpolation (SKI) .

$$\mathbf{K}_{\mathbf{X},\mathbf{X}} \approx \mathbf{K}_{\mathbf{X},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{X}} \approx \tilde{\mathbf{K}}_{\mathbf{X},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\tilde{\mathbf{K}}_{\mathbf{Z},\mathbf{X}} = \mathbf{W}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\mathbf{W} \triangleq \mathbf{K}_{\text{SKI}} \tag{2.89}$$

By exploiting the fast Kronecker matrix-vector multiplications mentioned above, the overall complexity of learning GP is $\mathcal{O}\left(DM^{1+\frac{1}{D}}\right)$ computations and $\mathcal{O}\left(N + DM^{\frac{2}{D}}\right)$ storage. Nonetheless, this approach also introduces additional design choices, such as determining the optimal density of the interpolation point grid, which require further fine-tuning than the relatively more straightforward inducing point methods. In general, the grid density is expected to be heavily dependent on the choice of the kernel since more expressive kernels are likely to require a greater number of interpolation points and less sparse $\mathbf{W}$. In summary, the combination of SKI and Kronecker algebraic structure results in the method KISS-GP.

2.4 Random Feature Approximations

As highlighted earlier, the inducing point-based approximation is a well-known approach for improving GPs' scalability. In these methods, a small number of pairs of inducing inputs and outputs are learned to define a new GPs, which is expected to be close as possible to the GPs, and the computational and storage cost now depend on the number of inducing points. These approaches are appropriate for locally complex functions. Intuitively, most inducing inputs would be located in regions where the function is complex, while the rest would be placed in regions where the function is simpler. Highly complex functions cannot be modeled well with these inducing point-based approaches.

In order to capture complex behaviors at a global level and improve the scalability of GPs, random feature-based approximations were proposed by Lázaro-Gredilla et al. (2010) and Gal and Turner (2015), which relies on spectral representations of kernel functions. For this kind of approximation, we

only consider stationary GPs whose covariance functions are written as a function of the distance between observations, i.e. $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}') = k(\mathbf{r})$. The spectral density for non-stationary kernels can be found in Remes et al. (2017). The concept of spectral expressiveness and random feature expansions are discussed here because these concepts are essential in the next chapter 2 where we propose a novel combination of CNNs and GPs approximated with random features.

2.4.1 Spectral Representations

Kernel trick and its problem. Kernel methods are a class of algorithms enabling the operations in an infinite-dimensional feature space, which leads to an enhancement of representational power. This is materialized by observing that inference for these methods is expressed through inner products between test points and input points, e.g. SVM (Cortes and Vapnik, 1995). Thanks to this observations and Mercer's theorem, we can implicitly define the transformation from the original space to the infinite-dimensional space by specifying the kernel function between points. This is the so-called *kernel trick*. However, the weakness of these methods is that algorithms needs to evaluate the kernel function between all pairs of datapoints. Consequently, large training sets incur large computational and storage costs.

Dual representation of a stationary kernel. Rahimi and Recht (2008) proposed an idea to define a transformation of the input space enabling a numerical approximation to kernel values without suffering a prohibitive cost. Due to the significant impact on research communities working on kernel-based models such as support vector machines, kernel ridge regression, and ultimately GPs, this seminal work is considered to be one of the most influential papers published in the previous decade. Their work is inspired by Bochner's theorem (Rudin, 1962) which states that any continuous shift-invariant normalized covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i - \mathbf{x}_j)$ is said to be positive definite if and only if it can be rewritten as the Fourier transform of some non-negative measure $p(\boldsymbol{\omega})$. The spectral density $s(\boldsymbol{\omega})$ can be constructed from $k(r)$ and vice versa through Wiener-Khintchin theorem:

$$k(\mathbf{r}) = \mathcal{F}^{-1}\{p(\boldsymbol{\omega})\} = \int_{-\infty}^{+\infty} p(\boldsymbol{\omega}) \exp\left(i\boldsymbol{\omega}^T\mathbf{r}\right) d\boldsymbol{\omega} \tag{2.90}$$

$$p(\boldsymbol{\omega}) = \mathcal{F}\{k(\mathbf{r})\} = \frac{1}{2\pi}\int_{-\infty}^{+\infty} k(r) \exp\left(-i\boldsymbol{\omega}^T\mathbf{r}\right) d\mathbf{r} \tag{2.91}$$

where $\mathcal{F}$ denotes Fourier transform and $e^{ix} = \cos x + i \sin x$ is the Euler's formula. Thanks to the relation indicated in Equation 2.91 and 2.90, several

examples of pairs of kernel function $k(r)$ and spectral density $p(\omega)$ can be given as follows:

Kernel Name	Kernel function $k(\mathbf{r})$	Spectral density $p(\boldsymbol{\omega})$
Gaussian	$\exp\left(-\frac{\|\|\mathbf{r}\|\|_2^2}{2}\right)$	$(2\pi)^{-\frac{D}{2}}\exp\left(-\frac{\|\|\boldsymbol{\omega}\|\|^2}{2}\right)$
Matérn 1/2	$\sigma^2\exp\left(-\frac{\|\|\mathbf{r}\|\|_1}{l}\right)$	$2\frac{\sigma^2}{l}\left(\frac{1}{l^2}+\|\|\boldsymbol{\omega}\|\|_2^2\right)^{-1}$
Laplacian	$\exp\left(-\|\|\mathbf{r}\|\|_1\right)$	$\prod_d^D \frac{1}{\pi\left(1+\omega_d^2\right)}$

Approximation of RBF Kernel using Random Fourier Features. Generally, we consider the RBF kernel parameterized by $\boldsymbol{\theta}=(\sigma^2,l_1,...,l_D)$ and its corresponding spectral density can be found using Equation 2.91 as follows:

$$k_{\text{RBF}}(\mathbf{x}_i,\mathbf{x}_j\mid\boldsymbol{\theta})=k_{\text{RBF}}(\mathbf{x}_i-\mathbf{x}_j\mid\boldsymbol{\theta})=k_{\text{RBF}}(\mathbf{r}\mid\boldsymbol{\theta})=\sigma^2\exp\left(\sum_{d=1}^{D}\frac{r_d^2}{l_d}\right). \tag{2.92}$$

$$p_{\text{RBF}}(\boldsymbol{\omega})=\mathcal{N}\left(\boldsymbol{\omega}\mid\mathbf{0},\boldsymbol{\Lambda}^{-1}\right)\ \text{, where } \boldsymbol{\Lambda}=\text{Diag}(l_1,\ldots,l_D). \tag{2.93}$$

From equation 2.90, the kernel function can be rewritten as the expectation under the density $p_{\text{RBF}}(\boldsymbol{\omega})$.

$$\begin{aligned}k_{\text{RBF}}(\mathbf{x}_i,\mathbf{x}_j\mid\boldsymbol{\theta})=k(\mathbf{r}\mid\boldsymbol{\theta})&=\sigma^2\text{E}_{p(\boldsymbol{\omega})}\left[\exp\left(i\boldsymbol{\omega}^T\mathbf{r}\right)\right]\\&=\sigma^2\text{E}_{p(\boldsymbol{\omega})}\left[\cos\left(\boldsymbol{\omega}^T\mathbf{r}\right)+i\sin\left(\boldsymbol{\omega}^T\mathbf{r}\right)\right].\end{aligned} \tag{2.94}$$

As $sin(.)$ is an odd function, i.e. $\sin(-x)=-\sin(x)$, the imaginary term can be canceled out from the expectation in Equation 2.94. Further, the kernel function can be approximated using N_{RF} spectral samples $\tilde{\boldsymbol{\omega}}$ from density function $p(\boldsymbol{\omega})$.

$$k_{\text{RBF}}(\mathbf{x}_i,\mathbf{x}_j\mid\boldsymbol{\theta})=k_{\text{RBF}}(\mathbf{r}\mid\boldsymbol{\theta})=\sigma^2\text{E}_{p_{\text{RBF}}(\boldsymbol{\omega})}\left[\cos\left(\mathbf{r}^T\boldsymbol{\omega}\right)\right]\approx\frac{\sigma^2}{N_{RF}}\sum_{r=1}^{N_{RF}}\cos\left(\mathbf{r}^T\tilde{\boldsymbol{\omega}}^{(r)}\right) \tag{2.95}$$

Replacing $\mathbf{r}$ by $\mathbf{x}_i-\mathbf{x}_j$ into equation 2.95, we can express the approximation of kernel function by an inner product representation:

$$k_{\text{RBF}}(\mathbf{x}_i,\mathbf{x}_j\mid\boldsymbol{\theta})\approx\frac{\sigma^2}{N_{RF}}\sum_{r=1}^{N_{RF}}\cos\left(\mathbf{x}_i^T\tilde{\boldsymbol{\omega}}^{(r)}-\mathbf{x}_j^T\tilde{\boldsymbol{\omega}}^{(r)}\right)=\phi_{\text{RBF}}(\mathbf{x}_i)^T\phi_{\text{RBF}}(\mathbf{x}_j), \tag{2.96}$$

where $\phi_{\text{RBF}}(\mathbf{x})$ is known as random features of $\mathbf{x}$ for RBF kernel, which is defined as follows:

$$\phi_{\text{RBF}}(\mathbf{x}) = \frac{\sigma^2}{N_{RF}}\Big[\cos\left(\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(1)}\right),\ldots,\cos\left(\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(N_{RF})}\right), \sin\left(\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(1)}\right),\ldots,\sin\left(\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(N_{RF})}\right)\Big]^T \tag{2.97}$$

Approximation of order-one ARC-COSINE Kernel using Random Features. In addition to working with RBF, we also consider order-one ARC-COSINE covariance which is a prevalent kernel function.

$$k_{\text{ARC}}(\mathbf{x}_i,\mathbf{x}_j \mid \boldsymbol{\theta}) = \frac{\sigma^2}{\pi}\left|\left|\Lambda^{-\frac{1}{2}}\mathbf{x}_i\right|\right|\left|\left|\Lambda^{-\frac{1}{2}}\mathbf{x}_j\right|\right|\left[\sin(\alpha)+(\pi-\alpha)\cos(\alpha)\right], \tag{2.98}$$

where $\boldsymbol{\theta} = (\sigma, \boldsymbol{\Lambda} = \text{Diag}(l_1^2,\ldots,l_D^2))$ and α is the angle between $\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{x}_i$ and $\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{x}_j$. Let $H(.)$ be the Heaviside function. Following Cho and Saul (2009), this covariance can be written under an integral form:

$$k_{\text{ARC}}(\mathbf{x}_i,\mathbf{x}_j \mid \boldsymbol{\theta}) = 2\sigma^2\int H\left(\boldsymbol{\omega}^T\mathbf{x}_i\right)\left(\boldsymbol{\omega}^T\mathbf{x}_i\right)H\left(\boldsymbol{\omega}^T\mathbf{x}_j\right)\left(\boldsymbol{\omega}^T\mathbf{x}_j\right)\times\mathcal{N}(\boldsymbol{\omega}\mid 0,\mathbf{I})\,d\boldsymbol{\omega}. \tag{2.99}$$

The convenient integral representation allows for a Monte Carlo approximation obtaining a low-rank approximation to the covariance matrix involving Rectified Linear Unit (ReLU) activation (Cho and Saul, 2009).

$$\phi_{\text{ARC}}(\mathbf{x}) = \sqrt{\frac{2\sigma^2}{N_{RF}}}\left[\max\left(0,\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(1)}\right),\ldots,\max\left(0,\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(N_{RF})}\right)\right]^T \tag{2.100}$$

2.4.2 Random featured-based Gaussian Processes.

In this section, I firstly present a well-known study of approximation of GPs using random features, which is proposed by Lázaro-Gredilla et al. (2010). The key novel idea is to sparsify the spectral representation of GPs.

Sparse Spectrum Gaussian Process Regression. As alluded earlier, Gaussian Processes Regression (GPR) is introduced in function-space view. Here, we remind that, by considering the dataset $\mathcal{D} = \{\mathbf{X},\mathbf{y}\}$ and Gaussian likelihood $p(\mathbf{y}\mid\mathbf{f}) = \mathcal{N}(\mathbf{y}\mid\mathbf{f},\sigma_n^2\mathbf{I})$, the predictive distributions $p(y_*\mid\mathbf{x}_*,\mathcal{D})$ and the logarithm of the marginal likelihood $\log(\mathbf{y}\mid\boldsymbol{\theta})$ given parameters $\boldsymbol{\theta}$ are expressed as follows:

$$\begin{aligned} p(y_*\mid\mathbf{x}_*,\mathcal{D}) &= \mathcal{N}\left(y_*\mid\mu_*,\sigma_*^2\right), \text{ where}\\ \mu_* &= \mathbf{K}_{\mathbf{x}_*,\mathbf{X}}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}}+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}\\ \sigma_*^2 &= \sigma_n^2 + \mathbf{K}_{\mathbf{x}_*} - \mathbf{K}_{\mathbf{x}_*,\mathbf{X}}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}}+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{K}_{\mathbf{X},\mathbf{x}_*} \end{aligned} \tag{2.101}$$

$$\log p\left(\mathbf{y} \mid \boldsymbol{\theta}\right) = -\frac{N}{2}\log\left(2\pi\right) - \frac{1}{2}\left|\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right| - \frac{1}{2}\mathbf{y}^T\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} \tag{2.102}$$

Computing the gradients of logarithm of the marginal likelihood with respect to related parameters requires the cubic cost to training size, i.e. $\mathcal{O}\left(N^3\right)$, which is unacceptable for large-scale data sets. In order to avoid the prohibitive cost, Lázaro-Gredilla et al. (2010) have employed the approximation of the covariance matrix using spectral representation. Consider for example ARD kernel (a stationary anisotropic squared exponential covariance function):

$$k_{\text{ARD}}\left(\mathbf{x}_i, \mathbf{x}_j \mid \boldsymbol{\theta}\right) = k_{\text{ARD}}\left(\mathbf{r} = \mathbf{x}_i - \mathbf{x}_j \mid \boldsymbol{\theta}\right) = \sigma_0^2 \exp\left(-\frac{1}{2}\mathbf{r}^T\boldsymbol{\Lambda}^{-1}\mathbf{r}\right), \tag{2.103}$$

where $\boldsymbol{\Lambda} = \text{Diag}\left(\left[l_1^2, \ldots, l_D^2\right]\right)$. Based on the dual representation of the stationary kernel mentioned above, we can approximate the $k_{\text{ARD}}\left(\mathbf{x}_i, \mathbf{x}_j\right)$ using N_{RF} spectral samples, and express the approximation as an inner product:

$$k_{\text{ARD}}\left(\mathbf{x}_i, \mathbf{x}_j\right) \approx \frac{\sigma_0^2}{N_{RF}}\sum_{r=1}^{N_{RF}}\cos\left(\mathbf{r}^T\tilde{\boldsymbol{\omega}}^{(r)}\right) = \frac{\sigma_0^2}{N_{RF}}\phi\left(\mathbf{x}_i\right)^T\phi\left(\mathbf{x}_j\right), \tag{2.104}$$

where $\tilde{\boldsymbol{\omega}}^{(r)} \sim p\left(\boldsymbol{\omega}\right) = \mathcal{N}\left(\boldsymbol{\omega} \mid 0, \boldsymbol{\Lambda}^{-1}\right)$, and we define $\phi(\mathbf{x})$ as a column vector of length $2N_{RF}$ containing the evaluation of the m pairs of trigonometric functions at $\mathbf{x}$.

$$\begin{aligned}\phi\left(\mathbf{x}\right) = \big[&\cos\left(\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(1)}\right), \ldots, \cos\left(\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(N_{RF})}\right), \\ &\sin\left(\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(1)}\right), \ldots, \sin\left(\mathbf{x}^T\tilde{\boldsymbol{\omega}}^{(N_{RF})}\right)\big]^T\end{aligned} \tag{2.105}$$

From the transformation $\phi\left(.\right)$, we construct $2N_{RF}$ by N matrix of random features $\boldsymbol{\Phi}_{\mathbf{X}} = \left[\phi\left(\mathbf{x}_1\right), \ldots, \phi\left(\mathbf{x}_N\right)\right]$. Now, the full kernel matrix $K_{\mathbf{X}}$ can be approximated as follows:

$$\mathbf{K}_{\mathbf{X}} \approx \frac{\sigma_0^2}{N_{RF}}\boldsymbol{\Phi}_{\mathbf{X}}^T\boldsymbol{\Phi}_{\mathbf{X}} \tag{2.106}$$

Replacing the kernel matrix by this approximation in equation 2.101 and 2.102, we obtain the spectral approximation of predictive distribution with mean μ_* and variance σ_*^2:

$$\mu_* \approx \phi\left(\mathbf{x}_*\right)\mathbf{A}^{-1}\boldsymbol{\Phi}_{\mathbf{X}}\mathbf{y}, \text{ and } \sigma_*^2 \approx \sigma_n^2 + \sigma_n^2\phi\left(\mathbf{x}_*\right)^T\mathbf{A}^{-1}\phi\left(\mathbf{x}_*\right), \tag{2.107}$$

where $\mathbf{A} = \boldsymbol{\Phi}_{\mathbf{X}}\boldsymbol{\Phi}_{\mathbf{X}}^T + \frac{N_{RF}\sigma_n^2}{\sigma_0^2}\mathbf{I}$. Similarly, we also obtain the approximate logarithm of the marginal likelihood:

$$\begin{aligned}\log p\left(\mathbf{y} \mid \boldsymbol{\theta}\right) \approx &-\left[\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\boldsymbol{\Phi}_{\mathbf{X}}^T\mathbf{A}^{-1}\boldsymbol{\Phi}_{\mathbf{X}}\mathbf{y}\right]/\left(2\sigma_n^2\right) - \frac{1}{2}\log\left|\mathbf{A}\right| \\ &+ N_{RF}\log\frac{N_{RF}\sigma_n^2}{\sigma_0^2} - \frac{N}{2}\log 2\pi\sigma_n^2\end{aligned} \tag{2.108}$$

Since this method approximates kernel matrices using the spectral density, it is called the Sparse Spectrum Gaussian Process (SSGP). Model selection can be done by optimizing jointly the logarithm of the marginal likelihood defined in 2.108 with respect to spectral points $\tilde{\boldsymbol{\omega}}^{(r)}$ and hyperparameters $\boldsymbol{\theta}$. The computational cost for each training step of SSGP algorithm is $\mathcal{O}\left(NN_{RF}^2\right)$. In terms of making prediction for each test point, the cost is $\mathcal{O}\left(N_{RF}\right)$ for the predictive mean and $\mathcal{O}\left(N_{RF}^2\right)$ for the predictive variance.

Extensions of SSGP. Gal and Turner (2015) show that the original SSGP model's have a tendency of overfitting. They have presented a Variational Sparse Spectrum approximation to the Gaussian Processes (VSSGP) that allows one to integrate out the set of spectral samples $\boldsymbol{\Omega} = \left[\tilde{\boldsymbol{\omega}}^{(1)}, \ldots, \tilde{\boldsymbol{\omega}}^{(N_{RF})}\right]$. The model VSSGP is shown to yield better calibrated uncertainty estimates accompanying predictions, and a procedure for deriving the optimal weights analytically is given for the Gaussian likelihood case. Other approaches of applying variational inference on SSGP are featured in Tan et al. (2015) and Hoang et al. (2016). Besides, efficient random feature maps have also been proposed to accelerate the computation and reduce the storage cost, such as the Fastfood approximation (Le et al., 2013) and Orthogonal Random Features (Yu et al., 2016).

2.5 Local Approximation

Inducing point-based and random feature-based approximations of GPs are implemented based on a global distillation, and they are commonly used to approximate GPs. However, these approaches require the computational and storage costs which are determined by auxiliary variables, i.e. number of inducing points or spectral samples. An alternative class of methods for improving the scalability of GPs is to follow the divide-and-conquer idea, which focuses on the local subsets of training data. According to the literature survey conducted by (Liu et al., 2018b), in this text, we opt to split the approach of local approximation into two groups: Separate-Local-Experts and Ensemble-Local-Experts.

Separate-Local-Experts. Intuitively speaking, there is almost no dependence between two points which are distant from each other. Thus, the prediction at an unseen input can be made sensibly by using localized experts with an acceptable computational cost. For example, Kim et al. (2005) and Datta et al. (2016) assume that a local expert model completely governs prediction at inputs inside its corresponding area. Simply, these approaches firstly parti-

tion the input space, then all local experts are trained based on these disjoint subsets, and then the inference at $\mathbf{x}_*$ can be made by an appropriate local expert. By introducing $\mathcal{M}_i$ as a local expert which is responsible for the subregion Ω_i and $\mathcal{D}_i$ as the subset of data located inside Ω_i, we mathematically state that the predictive distribution at $\mathbf{x}_*$ can be approximated by using a subset of data $\mathcal{D}_i$, i.e. $p(y_* \mid \mathcal{D}, \mathbf{x}_*) \approx p(y_i \mid \mathcal{M}_i, \mathcal{D}_i, \mathbf{x}_*)$. The partition on input space can be made by some clustering algorithms, e.g. Voronoi tessellations (Kim et al., 2005), and tree techniques (Vasudevan et al., 2009; Pratola et al., 2013). By restricting the number of data points of a local model to M, there are N/M local GPs where N is training size. Learning all independent GPs experts requires a cost of $\mathcal{O}(NM^2)$.

Instead of grouping data points into disjoint subsets statically before training local GPs experts, an alternative approach is to select a neighborhood subsets $\mathcal{D}_*$ around $\mathbf{x}_*$, and train a particular expert $\mathcal{M}_*$ to make the prediction at $\mathbf{x}_*$. For example, Urtasun and Darrell (2008) employ a dynamic partition to choose m_0 neighbor points around $\mathbf{x}_*$, resulting in $\mathcal{O}(n_t m_0^3)$ complexity that relies on the test size n_t. The primary problem of the approach is the concept of the neighborhood set $\mathcal{D}_*$ around $\mathbf{x}_*$. The most straightforward way is to use geometric closeness criteria for selection, i.e. the selected points should be close to $\mathbf{x}_*$. However, the approach is not optimal due to these closest points convey redundant information. Thus, there are several GP-based methods which have been employed to sequentially update the neighborhood set (Gramacy, 2016; Gramacy and Haaland, 2016; Gramacy and Lee, 2009; Gramacy and Apley, 2015).

While improving significantly the scalability and enjoying the capability of capturing non-stationary features due to the localized structure, Separate-Local-Experts yields discontinuous predictions on the boundaries of subregions, which is illustrated in Liu et al. (2018b). To alleviate the discontinuity problem, the patched GPs (Park and Huang, 2016; Park and Apley, 2018) restricts that two adjacent local GPs are patched to share the nearly identical predictions on the boundary. However, it possibly yields non-sensible predictive variances, and are only available in low dimensional space (Pourhabib et al., 2014b; Park and Apley, 2018). Another problem of Separable Local Experts is to suffer from poor generalization since it misses the long-term spatial correlations. To address the generalization issue, we can restrict that all local expert use the same hyperparameters (Deisenroth and Ng, 2015), or combine local and global approximation of GPs as mentioned in Snelson and Ghahramani (2007).

Ensemble-Local-Experts. An alternative solution to mitigate the problems raised by Separable Local Experts is to use the model averaging strategy, which is accomplished by an ensemble of local experts. The approach combines various local GPs possessing individual hyperparameters for enhancing accuracy and reliability (Yuksel et al., 2012a; Masoudnia and Ebrahimpour, 2014). Mathematically, Ensemble-Local-Experts can be expressed as a mixture of M Gaussian model, where the weight for each component can be seen as a gating function of covariates, which often takes a parametric form such as the softmax (Jacobs et al., 1991) and probit function (Geweke and Keane, 2007). More general, it can be extended to a tree-structured hierarchical architecture (Jordan and Jacobs, 1993).

The application of GPs mixture experts for big data scenarios must deal with various problems. For example, the question of determining the number of local experts can be dealt with by Akaike information criterion (Huang et al., 2014), or the synchronously balancing criterion (Zhao et al., 2015a). Another problem is on the reduction of computational cost, which includes several research directions. The first one is to the localization of experts. This can be accomplished by Expectation Maximization (ME) algorithm, wherein the data points are assigned to local experts through Maximum a Posterior in E-step (Nguyen and Bonilla, 2014b; Zhao et al., 2015b; Chen et al., 2014), and subsequently, the optimization in M-step only operates on small subsets of data. The second one is to combine global approximation with local experts. When using m inducing points for each local GPs that is responsible for n samples, the complexity for training M experts is intuitively $\mathcal{O}\left(nm^2M\right)$, which can be reduced to $\mathcal{O}\left(nm^2\right)$ using hard-cut EM (Nguyen and Bonilla, 2014b; Nguyen et al., 2016).

3
Calibrating Deep Convolutional Gaussian Processes

The wide adoption of Convolutional Neural Networks (CNNs) in applications where decision-making under uncertainty is fundamental, has brought a great deal of attention to the ability of these models to accurately quantify the uncertainty in their predictions. Previous work on combining CNNs with Gaussian processes (GPs) has been developed under the assumption that the predictive probabilities of these models are well-calibrated. In this paper we show that, in fact, current combinations of CNNs and GPs are miscalibrated. We propose a novel combination that considerably outperforms previous approaches on this aspect, while achieving state-of-the-art performance on image classification tasks.

3.1 Introduction

The wide adoption of Convolutional Neural Networks (CNNs) in increasingly popular pieces of technology such as self driving cars and medical imaging, where decision-making under uncertainty is fundamental, has brought attention to the ability of these learning architectures to accurately quantify the uncertainty in their predictions (Kendall and Gal, 2017; Gal and Ghahramani, 2016b). In short, the reliability of predictive probabilities of learning algorithms can be evaluated through the analysis of their calibration (Flach, 2016). In particular, a classifier is well calibrated when its output offers an accurate account of the probability of a given class, i.e. when it predicts a given class label with probability p that matches the true proportion p of test points belonging to that class.

The calibration properties of standard classifiers and neural networks have been studied in the literature (Kull et al., 2017; Niculescu-Mizil and Caruana,

2005), which has shown that classifiers that use the standard cross-entropy loss are generally well calibrated. Perhaps surprisingly, modern CNNs, which are a particular case of deep neural networks (DNNs), have been found to be miscalibrated, and the depth of convolutional filters is the main factor affecting calibration (Guo et al., 2017). The work in Guo et al. (2017) shows that regularization, implemented through weight decay, improves calibration and that, ultimately, simple methods such as post-calibration (Platt, 1999) can be an effective remedy for most calibration issues of CNNs.

Alternatively, Bayesian CNNs (Gal and Ghahramani, 2016b) where convolutional filters are inferred using Bayesian inference techniques, seem like perfect candidates to model uncertainty in these architectures in a principled way. However, while Bayesian CNNs have been shown to be effective in obtaining state-of-the-art performance in image classification tasks, we are not aware of studies that show their calibration properties. Hence, our first contribution is to investigate the calibration properties of Bayesian CNNs.

Along a similar vein, independently of the works on Bayesian CNNs, there have been other attempts to give a probabilistic flavor to CNNs by combining them with Gaussian processes (GPs, (Rasmussen and Williams, 2006)). Most of these approaches can be seen as a way to parameterize a CNN-based covariance for GPs, and the aim is to learn end-to-end both the filters and the GPs (see, e.g., Bradshaw et al. (2017); Wilson et al. (2016)). A crucial aspect that the literature has overlooked, however, is that methods that combine CNNs and GPs suffer from the same issues of miscalibration that characterize modern CNNs. Therefore, the second contribution of this paper is to show that current combinations of CNNs and GPs are miscalibrated.

Consequently, as our third contribution, we propose a novel combination of CNNs and GPs that is indeed well-calibrated, while being simple to implement. In particular, we propose to replace the fully connected layers of CNNs with GPs that we approximate with random features (Cutajar et al., 2017; Lázaro-Gredilla et al., 2010). Due to this approximation, the resulting model becomes a Bayesian CNN with a nonlinear transformation applied to the convolutional features. Building on the connection between variational inference and dropout, we apply Monte Carlo dropout (MCD, (Gal and Ghahramani, 2016a)) to carry out joint inference over the filters and the approximate GPs, thus obtaining an end-to-end learning method for the proposed model, which we call CNN+GP(RF). The resulting approach is characterized by a number of attractive features: (i) it is well calibrated, given that it uses the multinomial likelihood and the filters are regularized using Bayesian inference techniques;

(ii) it is as scalable as state-of-the-art CNNs, in so much as it can be trained using mini-batch updates and can exploit GPU and distributed computing; (iii) unlike other works that combine CNNs and GPs, it is as easy to implement as standard CNNs, as it leverages the equivalence of GPs approximated with random features and Bayesian DNNs (Cutajar et al., 2017; Gal and Turner, 2015; Neal, 1996), and the connections between dropout and variational inference (Gal and Ghahramani, 2016a). We extensively validate these properties in a variety of image classification tasks.

Our final contribution extends the above framework by replacing the last layer of CNNs with Deep GPs (Cutajar et al., 2017) and by proposing the use of structured random features to obtain faster and more compact GP approximations (Le et al., 2013; Yu et al., 2016). In all, our proposal considerably improves on classification accuracy compared to previous combinations of CNNs and GPs (e.g., ∼88% on CIFAR10 and ∼67% on CIFAR100, all without data augmentation), while being competitive with state-of-the-art CNNs; we are not aware of other GP works that approach these results. Crucially, we achieve these performance without compromising on calibration, again considerably improving on previous approaches that combine CNNs and GPs.

3.2 Related Work

Calibration of Convolutional Networks: The issue of calibration of classifiers in machine learning was popularized in the 90's with the use of support vector machines for probabilistic classification (Platt, 1999). Calibration techniques aim to learn a transformation of the output using a validation set in order for the transformed output to give a reliable account of the actual probability of class labels (Flach, 2016); interestingly, calibration can be applied regardless of the probabilistic nature of the untransformed output of the classifier. Popular calibration techniques include Platt scaling (Platt, 1999) and isotonic regression (Zadrozny and Elkan, 2002).

Classifiers based on Deep Neural Networks (DNNs) have been shown to be well-calibrated (Niculescu-Mizil and Caruana, 2005). The reason is that the optimization of the cross-entropy loss promotes calibrated output. The same loss is used in Platt scaling and it corresponds to the correct multinomial likelihood for class labels. Recent sudies on the calibration of CNNs, which are a particular case of DNNs, however, show that depth has a negative impact on calibration, despite the use of a cross-entropy loss, and that regularization improves the calibration properties of classifiers (Guo et al., 2017).

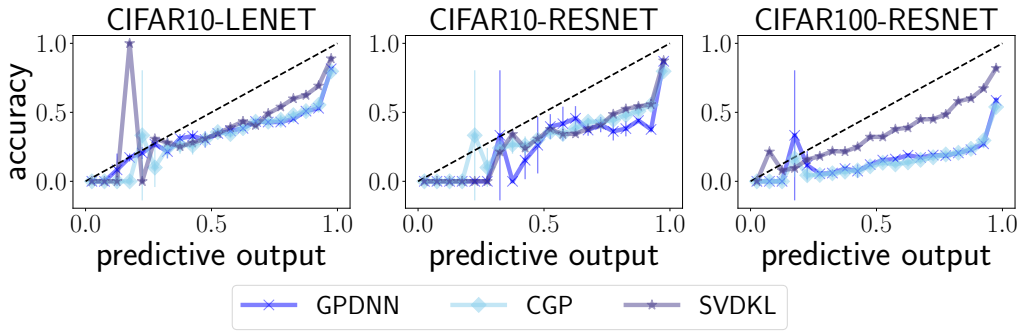


Figure 3.1 – Reliability diagrams for three state-of-the-art combinations of CNNs and GPs, i.e GPDNN (Bradshaw et al., 2017), CGP (van der Wilk et al., 2017), SVDKL (Wilson et al., 2016) applied to CIFAR10 and CIFAR100 data sets with LeNet and RESNET architectures. See table 3.1 for details on the convolutional architectures that we apply to CIFAR10 and CIFAR100. Because it is not possible to specify the convolutional structure in CGP (van der Wilk et al., 2017), the left and central panels show the same curve for CGP.

Combinations of Conv Nets and Gaussian Processes: Thinking of Bayesian priors as a form of regularization, it is natural to assume that Bayesian CNNs can "cure" the miscalibration of modern CNNs. Despite the abundant literature on Bayesian DNNs (Neal, 1996; Mackay, 1994), far less attention has been devoted to Bayesian CNNs (Gal and Ghahramani, 2016a), and the calibration properties of these approaches have not been investigated.

Several approaches have proposed the combination of CNNs and GPs as a means to give a probabilistic character to CNNs. Most of these works are based on ideas developed in the context of manifold GPs (Calandra et al., 2016), where inputs are transformed using some parametric transformation. In these works, the parametric transformation is based on convolutional layers, and scalability to large data is achieved through the use of ideas drawn from the literature on scalable GPs, for example the Stochastic Variational Deep Kernel Learning (SVDKL) approach in Wilson et al. (2016). In contrast, the work on hybrid GPs and DNNs (GPDNN, (Bradshaw et al., 2017)) combines CNNs and GPs using an inducing point approximation. Other recent approaches that aim to introduce convolutions in the calculation of the covariance between images include the work in van der Wilk et al. (2017), which proposes a way to construct covariances between domains/patches, mimicking the computations in CNNs.

In this work, we propose an alternative way to combine CNNs and GPs, where

GPs are approximated using random features expansions (Rahimi and Recht, 2008; Lázaro-Gredilla et al., 2010). The random feature expansion approximation amounts to replacing the orginal kernel matrix with a low-rank approximation, turning GPs into Bayesian linear models. Combining this with CNNs leads to a particular form of Bayesian CNNs, much like GPs and DGPs are particular forms of Bayesian DNNs (Duvenaud et al., 2014; Gal and Ghahramani, 2016a; Neal, 1996). Inference in Bayesian CNNs is intractable and requires some form of approximation. In this work, we draw on the interpretation of dropout as variational inference, employing the so-called Monte Carlo Dropout (MCD, (Gal and Ghahramani, 2016a)) to obtain a practical way of combining CNNs and GPs.

3.3 On calibration of Convolutional GPs

Consider a Q-class image classification task where $\mathbf{X}$ denotes a set of N images $\mathbf{x}_i \in \mathbb{R}^{p_x \times p_y} (1 \leq i \leq n)$, and $\mathbf{Y}$ is the matrix consisting of the corresponding one-hot encoded labels $\mathbf{y}_i$ stacked by row. We can use various metrics to determine the quality of a classifier, and here we focus in particular on calibration.

Let $\mathbf{g}(\mathbf{x})$ be the output of a classifier for an input image $\mathbf{x}$. To compute the calibration properties of a classifier, consider a partitioning of the test set $\mathbf{X}_*$ into disjoint sets $\{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$, such that each subset $\mathbf{X}_m$ contains the inputs yielding predictions in the range $(\frac{m-1}{M}, \frac{m}{M}]$. Hence, the confidence associated with each subset $\mathbf{X}_m$ is characterized by the midpoint of its corresponding range, i.e. $\text{conf}(\mathbf{X}_m) = \frac{m-0.5}{M}$. Then, the accuracy $\text{acc}(\mathbf{X}_m)$ for each subset can be evaluated as follows:

$$\frac{1}{|\mathbf{X}_m|} \sum_{\mathbf{x}_* \in \mathbf{X}_m} \delta \left(\arg\max(\mathbf{y}_*) - \arg\max(\mathbf{g}(\mathbf{x}_*)) \right), \tag{3.1}$$

where $\delta(x)$ is equal to one if $x = 0$, and zero otherwise.

In what follows, we use reliability diagrams to assess calibration, where we plot accuracy as a function of confidence for the subsets $\{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$. For a perfectly calibrated classifier, we expect $\text{acc}(\mathbf{X}_m) = \text{conf}(\mathbf{X}_m)$ for all m, with deviations implying that the class probabilities are either underestimated or overestimated. A useful summary statistics that can be extracted from reliability diagrams is the *Expected Calibration Error* (ECE), which is the average of the absolute difference between accuracy and confidence *weighted* according

to its size:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|\mathbf{X}_m|}{|\mathbf{X}_*|} \left|\text{acc}(\mathbf{X}_m) - \text{conf}(\mathbf{X}_m)\right|. \tag{3.2}$$

Another metric that measures the accuracy in predicting class probabilities is the BRIER score which takes into account the factors of calibration, resolution and uncertainty (Murphy, 1973). It is defined as the squared distance between labels and outputs averaged across classes and test points:

$$\text{BRIER} = \frac{1}{N_{\text{test}}} \sum_{\mathbf{x}_* \in \mathbf{X}_*} \frac{1}{Q} \sum_{k=1}^{Q} \left((\mathbf{y}_*)_k - (\mathbf{g}(\mathbf{x}_*))_k\right)^2. \tag{3.3}$$

In figure 3.1, we report the reliability diagrams of three state-of-the-art combinations of CNNs and GPs, i.e GPDNN approach in Bradshaw et al. (2017), CGP in van der Wilk et al. (2017) and SVDKL in Wilson et al. (2016). These approaches are applied to the CIFAR10 and CIFAR100 data sets with various convolutional structures. Note that the lines for CGP in the sub-figure of CIFAR10-LeNet and CIFAR10-RESNET are identical because there is no equivalent CNN architecture in CGP. All of reliability diagrams for these methods and ours can be found in the supplemental material.

The results indicate that current approaches that combine CNNs and GPs are miscalibrated, with a tendence of being overconfident in predictions. This is an important and perhaps surprising finding, because one of the motivations to combine CNNs with GPs is to do better quantification of uncertainty compared to plain CNNs. In the experiments section we report more extensively on the calibration of these classifiers, as well as illustrating other performance metrics. These considerations call for the study of better ways to combine CNNs and GPs to recover calibration while attempting to improve on standard metrics such as error rate and test log-likelihood. The next section illustrates our proposal that achieves this goal.

3.4 Proposed Method

In the proposed model, the labels $\mathbf{Y}_{i\cdot}$ are assumed to be conditionally independent given a set of corresponding latent variables $\mathbf{F}_{i\cdot}$, i.e. we consider the likelihood $p(\mathbf{Y}|\mathbf{F}) = \prod_{i=1}^{N} p(\mathbf{Y}_{i\cdot}\ |\mathbf{F}_{i\cdot})$, where the latent variables $\mathbf{F}$ are realizations of a set of Q functions $f_j(\mathbf{x})$ at the input images $\mathbf{x}_1, \ldots, \mathbf{x}_n$, i.e., $(\mathbf{F})_{ij} = f_j(\mathbf{x}_i)$ for $j = 1, \ldots, Q$. Each individual $p(\mathbf{Y}_{i\cdot}\ |\mathbf{F}_{i\cdot})$ is multinomial with probabilities obtained using a softmax transformation of the latent variables. In this work we focus on functions $f_j(\mathbf{x})$ that are modeled using GPs;

note that extension to DGPs is actually easy to consider in our framework, as we show in the experiments.

Due to the GP modeling assumption, the latent function values $\mathbf{F}_{\cdot j}$ comprising $(f_j(\mathbf{x}_1), \ldots, f_j(\mathbf{x}_n))^\top$ are jointly Gaussian with $p(\mathbf{F}_{\cdot j}|\mathbf{X}, \boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where $\mathbf{K}$ is the covariance matrix. The entries of the covariance matrix $\mathbf{K} = \{k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta})\}_{i,j}$, are specified by a covariance (kernel) function k (with hyperparameters $\boldsymbol{\theta}$) and this form is shared across output dimensions, although this can be relaxed and allow for a different k for the Q outputs.

Instead of applying the GP modeling directly to the images, we propose to employ a transformation $\mathbf{c}(\mathbf{x}|\boldsymbol{\Psi})$ using convolutional layers, where $\boldsymbol{\Psi}$ denotes the parameters of such layers. The vector-valued function $\mathbf{c}(\mathbf{x}|\boldsymbol{\Psi})$ is differentiable as it implements a series of differentiable operations, such as convolutions and pooling. This is one of the key successes of CNN models that allows for the learning of their filters, which we exploit for the end-to-end learning of our model.

Inference in this model requires being able to characterize the posterior over all or a selected group of model parameters, but this posterior is analytically intractable and thus computationally prohibitive (Rasmussen and Williams, 2006). In the remainder of this paper, we build on previous work on scalable inference for GPs and DGPs with random features (Cutajar et al., 2017) to obtain an approximation to the proposed model that can be learned end-to-end.

3.4.1 Random Feature Expansions

Naïve inference in GP models requires algebraic operations with $\mathbf{K}$ that would cost $\mathcal{O}(n^3)$ in time. Popular approaches to recover tractability use low-rank approximations of the kernel matrix. Among this family of low-rank approximations, we choose to work with random feature approximations (Lázaro-Gredilla et al., 2010; Cutajar et al., 2017). The reason is that they offer a number of possible extensions to speedup computations (e.g., using structured approximations (Le et al., 2013; Yu et al., 2016)) and increase the complexity of the model (e.g., considering Deep GPs (Cutajar et al., 2017)); we elaborate on this in the experiments section. In random feature expansions, the kernel matrix is replaced by a low-rank approximation $\mathbf{K} \approx \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$, with $\boldsymbol{\Phi} \in \mathbb{R}^{n \times m}$ and $m \ll n$. This approximation suggests the construction of a Bayesian linear model to approximate the GP latent variables as $\mathbf{F} = \boldsymbol{\Phi}\mathbf{W}$. Using $p(W_{ij}) = \mathcal{N}(W_{ij}|0, 1)$ it is straightforward to show that the covariance of each of the latent functions $\mathbf{F}_{\cdot j}$ is indeed an approximation to $\mathbf{K}$, as

$\text{cov}(\mathbf{F}_{\cdot j}) = \text{E}(\mathbf{\Phi}\mathbf{W}_{\cdot j}\mathbf{W}_{\cdot j}^\top\mathbf{\Phi}^\top) = \mathbf{\Phi}\text{E}(\mathbf{W}_{\cdot j}\mathbf{W}_{\cdot j}^\top)\mathbf{\Phi}^\top = \mathbf{\Phi}\mathbf{\Phi}^\top \approx \mathbf{K}.$

In this work, we focus in particular on the order-one ARC-COSINE kernel (Cho and Saul, 2009)

$$k_{\text{arc}}^{(1)}(\mathbf{x}_i, \mathbf{x}_j|\mathbf{\Psi}, \boldsymbol{\theta}) = \frac{\sigma^2}{\pi}\left\|\Lambda^{-\frac{1}{2}}\mathbf{c}(\mathbf{x}_i|\mathbf{\Psi})\right\|\left\|\Lambda^{-\frac{1}{2}}\mathbf{c}(\mathbf{x}_j|\mathbf{\Psi})\right\| \left[\sin(\alpha) + (\pi - \alpha)\cos(\alpha)\right], \tag{3.4}$$

where $\boldsymbol{\theta} = (\sigma, \mathbf{\Lambda} = \text{Diag}(\ell_1^2, \ldots, \ell_d^2))$ and α is the angle between $\Lambda^{-\frac{1}{2}}\mathbf{c}(\mathbf{x}_i|\mathbf{\Psi})$ and $\Lambda^{-\frac{1}{2}}\mathbf{c}(\mathbf{x}_j|\mathbf{\Psi})$.

The ARC-COSINE covariance has a convenient integral representation that allows for a Monte Carlo approximation, obtaining a low-rank approximation to the covariance matrix involving Rectified Linear Unit (RELU) activations (Cho and Saul, 2009)

$$\mathbf{\Phi}_{\text{arc}} = \sqrt{\frac{2\sigma^2}{N_{\text{RF}}}}\max\left(\mathbf{0}, \mathbf{C}(\mathbf{X}|\mathbf{\Psi})\,\mathbf{\Omega}\right). \tag{3.5}$$

In this expression, we have defined $\mathbf{C}(\mathbf{X}|\mathbf{\Psi})$ as the matrix resulting from the application of convolutional layers to the image training set $\mathbf{X}$ and $\mathbf{\Omega}$ is obtained by stacking N_{RF} samples from $p(\boldsymbol{\omega}) = \mathcal{N}\left(\boldsymbol{\omega}|\mathbf{0}, \mathbf{\Lambda}^{-1}\right)$ by column. Note that in the case of a popular Radial Basis Function (RBF) covariance, it is possible to obtain a similar random feature approximation, where the RELU activation is replaced by trigonometric functions; see Rahimi and Recht (2008) and the supplement for details.

3.4.2 End-to-end learning

Inference in the proposed model is intractable due to the likelihood that is not conjugate to the GP prior. Further complications stem from the need to infer kernel parameters, which include convolutional parameters, and the need to be able to scale to large data. Our aim is to carry out inference within a consistent framework that is characterized by simplicity, as described next.

We start by introducing an approximate posterior over $\mathbf{W}, \mathbf{\Omega}$ and $\mathbf{\Psi}$, that we denote as $q(\mathbf{W}, \mathbf{\Omega}, \mathbf{\Psi})$. Following standard variational inference arguments, we can define an operative way to obtain these approximate posteriors. The log-marginal likelihood $\mathcal{L} = \log\left[p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}\right]$ can be bounded by the sum of an expected log-likelihood term and a negative Kullback-Leibler (KL) divergence

term as follows:

$$\begin{aligned}\mathcal{L} \geq & \mathrm{E}_{q(\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Psi})}\left(\log\left[p\left(\mathbf{Y}|\mathbf{X},\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Psi},\boldsymbol{\theta}\right)\right]\right) \\ & - \mathrm{KL}\left[q\left(\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Psi}\right)\|p\left(\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Psi}\right)\right].\end{aligned} \tag{3.6}$$

Variational inference amounts to optimizing the lower bound above with respect to $q(\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Psi})$ and any other parameters of interest.

We have now a number of options on the form for the approximate posteriors $q(\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Psi})$. In previous works on variational inference for DNNs, it has been proposed to define the approximating distributions to be Gaussian and factorized across parameters (Kingma and Welling, 2014; Graves, 2011). The drawback of this is that it doubles the number of parameters. Alternatively, we can rely on the connections between dropout and variational inference (Gal and Ghahramani, 2016a,b) which is drawn by assuming the posterior of $\mathbf{W},\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}$ as a mixture of two Gaussian distributions (see supplement). From this connection, we are able to obtain an easier approximate inference scheme, which is also known as Monte Carlo Dropout (MCD). Focusing on the weights for now, the connection with dropout is apparent if we rewrite

$$\mathbf{W} = \mathbf{M}_w \ \mathrm{Diag}[\mathbf{z}_w] \tag{3.7}$$

with $(\mathbf{z}_w)_i \sim \mathrm{Bernoulli}(\pi_w)$. The reparameterization introduces variational parameters $\mathbf{M}_w$ (one for each weight in $\mathbf{W}$) and a vector of binary variables that can switch on or off the columns of the weight matrix with probability π_w. A similar reprameterization can be done for the convolutional parameters $\boldsymbol{\Psi}$ and matrices of random feature $\boldsymbol{\Omega}$, introducing $\mathbf{M}_\psi, \mathbf{M}_\Omega$ and π_ψ, π_Ω. The optimization of the lower bound wrt all variational parameters requires being able to evaluate the expectation and the KL term in (3.16).

In MCD, the KL term in (3.16) can be approximated following Gal and Ghahramani (2016a), obtaining a regularization term involving the squared-norm of the parameters

$$\mathrm{KL}\left[q\left(\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Psi}\right)\|p\left(\mathbf{W},\boldsymbol{\Omega},\boldsymbol{\Psi}\right)\right] \ \approx \ \frac{\pi_w}{2}\left\|\mathbf{M}_w\right\|^2 + \frac{\pi_\Omega}{2}\left\|\mathbf{M}_\Omega\right\|^2 + \frac{\pi_\psi}{2}\left\|\mathbf{M}_\psi\right\|^2 \tag{3.8}$$

The expectation in (3.16), instead, can be unbiasedly estimated using Monte Carlo and also considering a mini-batch of size m:

$$\frac{N}{m}\frac{1}{N_{\mathrm{MC}}}\sum_{i=1}^{N_{\mathrm{MC}}}\sum_{k\in\mathcal{I}_m}\log\left[p\left(\mathbf{y}_k|\mathbf{x}_k,\mathbf{W}^{(i)},\boldsymbol{\Omega}^{(i)},\boldsymbol{\Psi}^{(i)},\boldsymbol{\theta}\right)\right] \tag{3.9}$$

with $\mathbf{W}^{(i)}, \mathbf{\Omega}^{(i)}, \mathbf{\Psi}^{(i)} \sim q(\mathbf{W}, \mathbf{\Omega}, \mathbf{\Psi})$, and $\mathcal{I}_m$ is a set of m indices to select a mini-batch of training points (Graves, 2011). This doubly-stochastic approximation is differentiable wrt variational parameters when the Bernoulli variables are fixed.

The approximate objective can now be optimized in the same vein as in standard back-propagation with dropout, noting that dropout is applied to $\mathbf{W}$, $\mathbf{\Omega}$ and to convolutional parameters $\mathbf{\Psi}$. What changes, however, is the interpretation of the procedure as stochastic variational inference, whereby the Bernoulli variables are resampled at each iteration. A practical implication is in the way we compute the predictive distribution, which has a probabilistic flavor as follows:

$$p(\mathbf{y}_*|\mathbf{x}_*, X, \boldsymbol{\theta}) \approx \int p(\mathbf{y}_*|\mathbf{W}, \mathbf{\Omega}, \mathbf{\Psi}, \mathbf{x}_*, X, \boldsymbol{\theta}) q(\mathbf{W}, \mathbf{\Omega}, \mathbf{\Psi}) d\mathbf{W} d\mathbf{\Omega} d\mathbf{\Psi}, \quad (3.10)$$

and can be approximated using Monte Carlo by resampling the Bernoulli variables. While MCD has been proposed for CNNs in (Gal and Ghahramani, 2016b), in this work we extend it to the case of joint inference over convolutional parameters and the GP approximation in the CNN+GP(RF) model, thus obtaining a practical inference and prediction scheme, which combines CNNs and GPs.

Depth	Data set	CNN architecture	CNN name
Shallow	MNIST	2 Conv Layers + 2 Fully connected	LeNet
Shallow	CIFAR10	2 Conv Layers + 3 Fully connected	LeNet
Deep	CIFAR10	30 Conv Layers + 1 Fully connected	RESNET
Deep	CIFAR100	150 Conv Layers + 1 Fully connected	RESNET

Table 3.1 – CNN architectures considered in this work. The same architectures are used in GPDNN and SVDKL by replacing the fully connected layers with GPs, while CGP does not explicitly use a convolutional structure.

3.4.3 Extensions

Structured random feature approximations: One of the advantages of the proposed model, compared to other GP approximations, is that it can exploit structured random feature expansions to accelerate computations and reduce the size of the approximate GP (Le et al., 2013; Yu et al., 2016). In the random features approximation, random features are constructed by multiplying $\mathbf{\Omega}$ with the convolutional features. Without loss of generality, assuming that $\mathbf{\Omega} \in \mathbb{R}^{m \times d}$ and $\mathbf{c}(\mathbf{x}|\mathbf{\Psi}) \in \mathbb{R}^{d \times 1}$, the cost of computing products $\mathbf{\Omega}\mathbf{c}(\mathbf{x}|\mathbf{\Psi})$ is $\mathcal{O}(md)$, while storing $\mathbf{\Omega}$ requires $\mathcal{O}(md)$ storage.

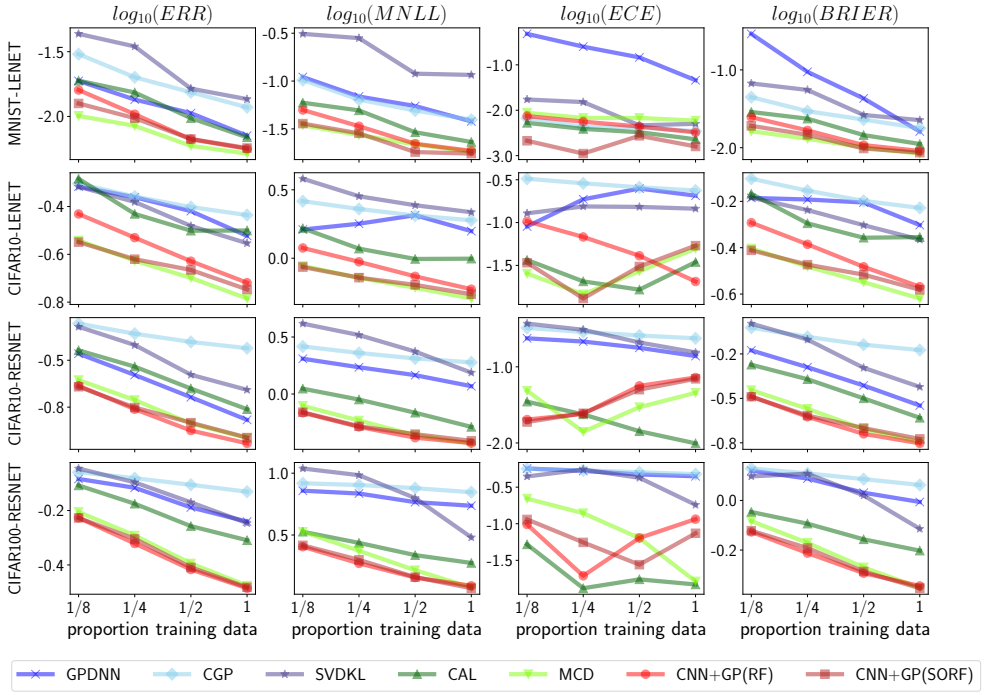


Figure 3.2 – Comparison of our CNN+GP(RF) and CNN+GP(SORF) with existing combinations of CNNs with GPs, and with Bayesian CNNs and post-calibrated CNNs. All performace metrics are defined so that the lower the better.

Structured approximations aim to reduce the time complexity to $\mathcal{O}(m \log d)$ and the storage cost to $\mathcal{O}(m + d)$. Taking a standard random features expansion of the isotropic covariance in (3.5) with $\mathbf{\Lambda} = \ell^{-2}\mathbf{I}$ as an example, $\mathbf{\Omega} = \frac{1}{\ell}\mathbf{G}$, with $\mathbf{G}_{ij} \sim \mathcal{N}(0, 1)$. One way to make computations cheaper is to replace the Gaussian matrix $\mathbf{G}$ with a pseudo-random alternative. The Structured Orthogonal Random Feature (SORF) approximation (Yu et al., 2016) approximates $\mathbf{G}$ through a series of Hadamard transformations of diagonal matrices $\mathbf{D}_i$ with elements randomly sampled from $\{-1, +1\}$ or Rademacher distribution, that is $\mathbf{G} \approx \sqrt{d}\mathbf{H}\mathbf{D}_1\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_3$, where $\mathbf{H}$ is the normalized Walsh-Hadamard matrix. We refer to this variation of the model as CNN+GP(SORF).

Similarly to the other parameters, we infer the diagonal matrices $\mathbf{D}_i$ using MCD. We denote by $\mathbf{d}_i$ the diagonal of $\mathbf{D}_i, i = 1, 2, 3$. The MCD scheme (Gal and Ghahramani, 2016a,b) assumes an L_2 regularization which implies a zero-mean Gaussian prior, which is inappropriate for $\mathbf{d}_i$ as it is Rademacher distributed. We propose to bypass this limitation by applying MCD to a reparameterization of $\mathbf{d}_i$. In particular, denoting by $\mathbf{d}_i^* \in \{-1, +1\}^d$ the initialized values of $\mathbf{d}_i$, we apply MCD to $\mathbf{d}_i - \mathbf{d}_i^*$. According to this choice,

each diagonal element is sampled based on the variational parameters $\mathbf{M}_{\mathbf{d}_i - \mathbf{d}_i^*}$

$$\mathbf{d}_i = \begin{cases} \mathbf{M}_{\mathbf{d}_i - \mathbf{d}_i^*} + \mathbf{d}_i^*, & \text{with probability } \pi_d \\ \mathbf{d}_i^*, & \text{otherwise} \end{cases} \tag{3.11}$$

Convolutional Networks with Random-Feature-Expanded Deep GPs: A DGP model represents a deep probabilistic nonparametric approach where the output of one GP at each layer is used as the input to the GP in the next layer (Damianou and Lawrence, 2013). Extending the random feature approximation to DGPs and the inference scheme presented here is straightforward; see Cutajar et al. (2017) for details. The random feature approximation turns the DGP into a Bayesian DNN for which we can apply stochastic variational inference to infer model parameters. In the experiments section, we explore the possibility to stack a DGP on top of convolutional layers, and we show the impact of depth on performance.

3.5 Experiments

We carry out the experimental evaluation using popular benchmark datasets, such as MNIST, CIFAR10 and CIFAR100 and with a number of popular CNN architectures based on LeNet and RESNET (see table 3.1).

We report three state-of-the-art competitors combining CNNs and GPs, namely GPDNN (Bradshaw et al., 2017), SVDKL (Wilson et al., 2016), and CGP (van der Wilk et al., 2017). We also report Bayesian CNNs, as suggested in Gal and Ghahramani (2016b) and CNNs with post-calibration as proposed in Guo et al. (2017), which we refer to as CNN+MCD and CNN+CAL, respectively. For all the competing methods we used available implementations, adding the same CNN architecture to ensure a fair comparison. In all experiments, we use a batch-size $m = 100$ and the Adam optimizer with default learning rate (Kingma and Ba, 2015). In the methods that use MCD, we use a dropout rate of 0.5 for all parameters.

The results are reported in figure 3.2, where we have used different training sizes N, keeping the classes balanced. In the figure, we report the calibration measures that we have introduced earlier, namely ECE and BRIER scores, and we also report the classification error rate (ERR) and the mean negative test log-likelihood (MNLL). Compared to other combinations of CNNs and GPs, CNN+GP(RF) improves considerably on all metrics. It is interesting to see that our proposal is competitive with Bayesian CNNs employing MCD, with only a marginal improvement on ERR and MNLL in some configurations.

In TEMP it is necessary to leave out part of the data to perform post-calibration, which can be problematic in applications where obtaining labeled data is difficult or expensive. As a result, our proposal is considerably better, although TEMP is competitive in ECE; this is expected given that this is the metric that is optimized after training.

The two variants of our approach, namely CNN+GP(RF) where we learn the frequencies $\boldsymbol{\Omega}$ and CNN+GP(SORF) where we sample $\boldsymbol{\Omega}$ from its prior, are comparable. This suggests that the extra level of complexity of learning the spectral frequencies does not lead to substantial gains in performance and that the structured random feature approximation yields satisfactory performance.

We also note that these results have been obtained by fixing the covariance parameters $\boldsymbol{\theta}$ of the GP, as we found it to be unstable when learning these jointly with $\boldsymbol{\Omega}$. This might be the reason why these parameters were learned through cross-validation in Gal et al. (2017). In the supplement, we report the results obtained when learning $\boldsymbol{\theta}$ and fixing $\boldsymbol{\Omega}$, which we found yielding similar performance as fixing $\boldsymbol{\theta}$. All these observations corroborate the hypothesis that most of the performance of CNN-based classification models is due to the convolutional layers.

In summary, figure 3.2 shows that our CNN+GP(RF) is the best strategy for calibrating these models compared to other approaches using GPs. Furthermore, we found perhaps surprisingly that MCD has comparable performance. In the supplementary material, we report results on GPDNN where we infer convolutional parameters using MCD, so as to gain insights as to whether most of the improvements in performance are due to this form of regularization. The results support the intuition that inferring these parameters yields improvements in calibration, but also that our CNN+GP(RF) still offers better performance.

3.5.1 Reliability diagrams

In figure 3.3, we report the reliability diagrams of all the methods studied in figure 3.1. The figure shows that TEMP, MCD and CNN+GP(RF) produce well-calibrated predictions when using a shallow convolutional structure (LeNet). For a deeper architecture (RESNET), CNN+GP(RF) is slightly under-confident. Compared to previous combinations of CNNs and GPs, our approach yields better reliability curves.

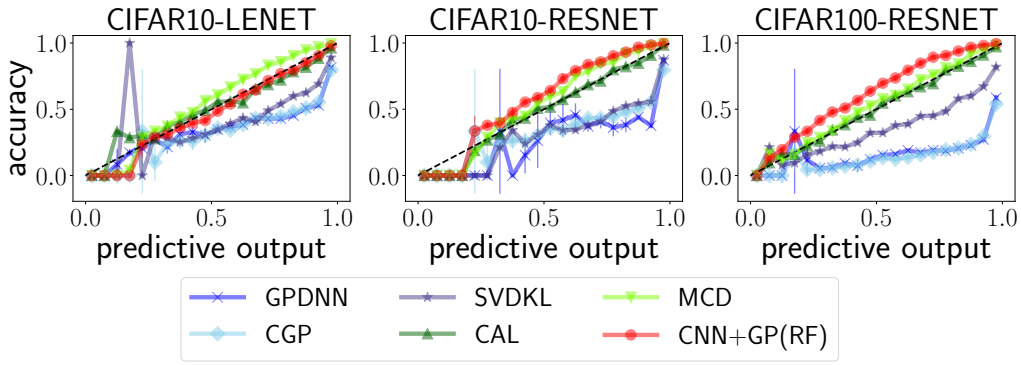


Figure 3.3 – Reliability diagrams of our CNN+GP(RF) in comparison with existing combinations of CNNs with GPs, and with Bayesian CNNs and post-calibrated CNNs.

3.5.2 Extension with Deep GPs

In figure 3.4, we report results varying the depth of a DGP on top of the convolutional layers; again, we learn the convolutional filters and the DGP end-to-end as discussed in the previous sections. We show results when applying our model to the whole CIFAR10 data set in the case of the shallow convolutional structure (table 3.1). We feed-forward the convolutional features to all layers of the DGP, in line with what suggested in the literature of DGPs to avoid pathologies in the functions that can be modeled (Cutajar et al., 2017; Duvenaud et al., 2014; Neal, 1996). The results indicate that increasing the complexity of the model improves on all performance metrics, and worsen calibration, which however is still around 3% ECE. This is in line with the intuition that increasing model complexity negatively impacts calibration.

3.5.3 Knowing when the model does not know

We report experiments showing the ability of our model to know when it does not know, following a similar experimental setup as in Lakshminarayanan et al. (2017). In this experiment we train our CNN+GP(RF) model on MNIST and test on the NOT-MNIST dataset, which contains images of letters from "A" to "J" in various typefaces. For this experiment, while we do not know the exact value that we should obtain for predictive probabilities, we expect to observe low entropy in the predictions when tesing on MNIST and high entropy when predicting on NOT-MNIST, indicating high uncertainty. The results are reported in figure 3.5, where we show the cumulative distribution of the entropy of predictive probabilities for two depths of the convolutional structure. In the figure, we compare our CNN+GP(RF) against one of the

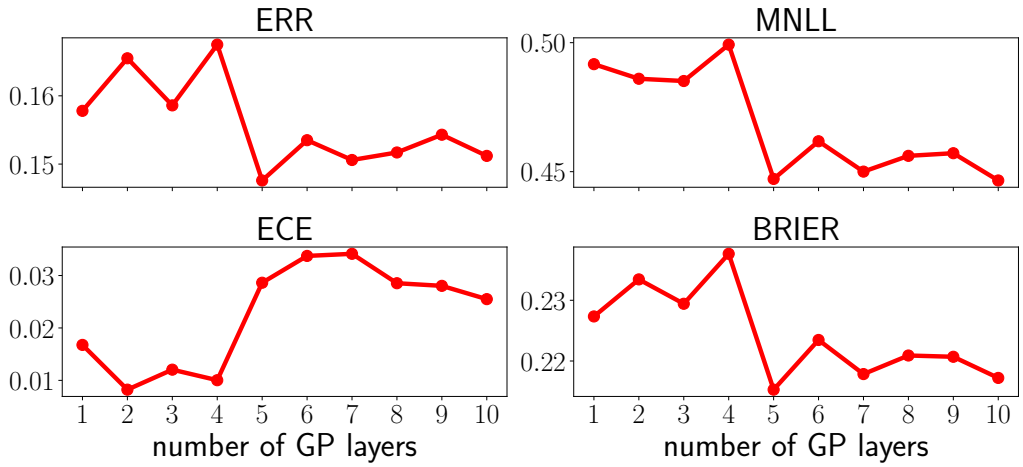


Figure 3.4 – Performance of the proposed model when varying the depth of the DGP on top of a RESNET convolutional structure on CIFAR10 dataset. Note that the scale of y-axes indicates that the metrics change only slightly when increasing the depth of the DGP.

methods combining CNNs and GPs, that is GPDNN. In the figure, we also include results on CNNs with post-calibration and Bayesian CNNs inferred with MCD. Our approach is competitive with Bayesian CNNs and it is considerably superior to post-calibration. This is especially true in the case of the RESNET convolutional structure, where post-calibration still yields a large number of predictions with low uncertainty. Interestingly, GPDNN assigns large uncertainty to predictions on NOT-MNIST, although with the deeper convolutional architecture it yields a large fraction of predictions with low entropy. We speculate that this due to the inducing point approximation of the GP, which nicely captures uncertainty away from training data except for test points which are closer to the training data.

3.5.4 Extension with the SORF

In table 3.2, we report further results comparing MCD with CNN+GP(SORF). In this experiment, we use the ALEXNET structure (Krizhevsky et al., 2012) on CIFAR10 and CIFAR100 datasets. The results in table 3.2 show improvements in using our model compared CNNs with MCD. We attribute this to the fact that the GP approximated through SORF in place of the fully connected layer of ALEXNET reduces model parameters from 30 million to 2.3 million.

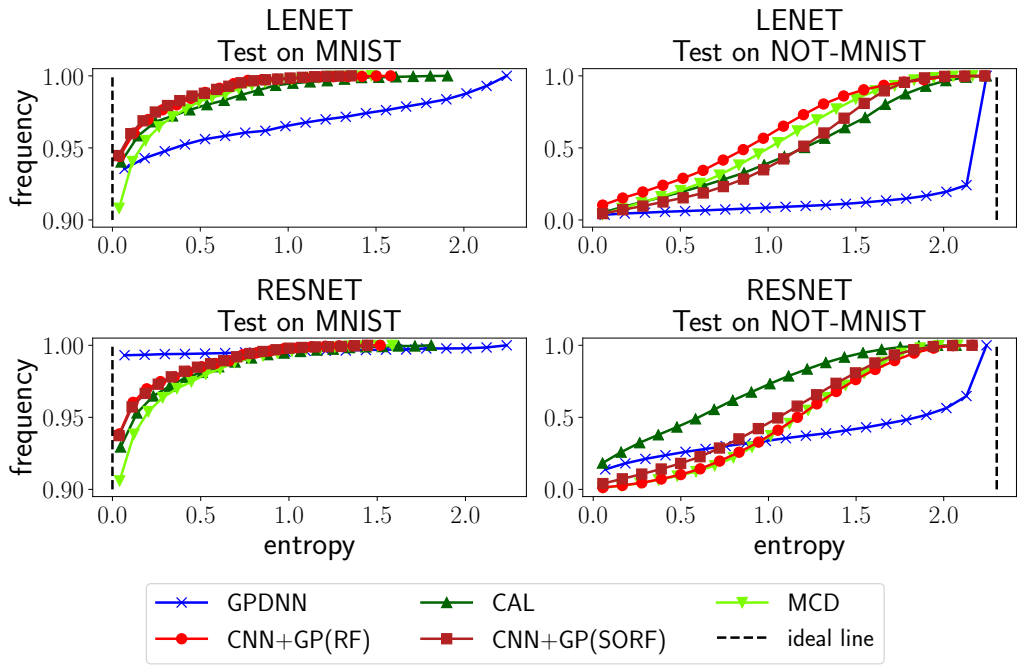


Figure 3.5 – Cumulative distribution function plot of predictive entropies when the models trained on MNIST are tested on MNIST and NOT-MNIST. We report results for two different depths of the convolutional structure. NOT-MNIST dataset available at `http://yaroslavvb.blogspot.fr/2011/09/notmnist-dataset.html`

METHOD	Dataset	ERR	MNLL	ECE	BRIER
CNN+GP(SORF)	CIFAR10	0.172	0.522	0.063	0.250
MCD	CIFAR10	0.181	0.591	0.110	0.276
CNN+GP(SORF)	CIFAR100	0.459	1.806	0.127	0.612
MCD	CIFAR100	0.594	2.434	0.058	0.732

Table 3.2 – Comparison between CNN+GP(SORF) and MCD with ALEXNET architecture on CIFAR10 and CIFAR100.

3.6 Mathematical details and other experiments

3.6.1 Random Feature Expansion of the RBF Covariance

We report here the expansion of the popular Radial Basis Function (RBF) covariance. Following the convolutional representation of images in our CNN+GP(RF) model, the RBF covariance is defined as:

$$k_{\text{rbf}}(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\Psi}, \boldsymbol{\theta}) = \sigma^2 \exp\left[-\left(\mathbf{c}(\mathbf{x}_i|\boldsymbol{\Psi}) - \mathbf{c}(\mathbf{x}_j|\boldsymbol{\Psi})\right)^\top \boldsymbol{\Lambda}^{-1} \left(\mathbf{c}(\mathbf{x}_i|\boldsymbol{\Psi}) - \mathbf{c}(\mathbf{x}_j|\boldsymbol{\Psi})\right)\right], \tag{3.12}$$

with $\boldsymbol{\theta} = (\sigma, \boldsymbol{\Lambda} = \text{Diag}(\ell_1^2, \ldots, \ell_d^2))$. It is possible to express this covariance function as the Fourier transform of a non-negative measure $p(\boldsymbol{\omega})$ Rahimi and Recht (2008), where $\boldsymbol{\omega}$ are the so-called spectral frequencies. It is straightforward to verify that $p(\boldsymbol{\omega}) = \mathcal{N}\left(\boldsymbol{\omega}|\mathbf{0}, \boldsymbol{\Lambda}^{-1}\right)$. Stacking N_{RF} Monte Carlo samples from $p(\boldsymbol{\omega})$ into $\boldsymbol{\Omega}$ by column, we obtain

$$\boldsymbol{\Phi}_{\text{rbf}} = \sqrt{\frac{\sigma^2}{N_{\text{RF}}}} \left[\cos\left(\mathbf{C}(\mathbf{X}|\boldsymbol{\Psi})\,\boldsymbol{\Omega}\right), \sin\left(\mathbf{C}(\mathbf{X}|\boldsymbol{\Psi})\,\boldsymbol{\Omega}\right)\right], \tag{3.13}$$

where $\mathbf{C}(\mathbf{X}|\boldsymbol{\Psi})$ denotes the matrix resulting from the application of convolutional layers to the image training set $\mathbf{X}$, and the sin and cos functions are applied elementwise to their argument.

3.6.2 Variational Inference for the Proposed Model

3.6.2.1 CNN+GP(RF)

In CNN+GP(RF), the variational parameters we would like to optimize are $\mathbf{M}_w, \mathbf{M}_\psi$ and $\mathbf{M}_\Omega$. Our model parameters $\mathbf{W}, \boldsymbol{\Psi}$ and $\boldsymbol{\Omega}$ share an identical form for the approximate posterior and prior. Focusing on $\mathbf{W}$, its elements have a standard normal prior, and we assume that the posterior $q(\mathbf{W})$ is a mixture of two Gaussian distribution, which can be factorized over rows, governed by variational parameters $\mathbf{M}_w$:

$$q(\mathbf{W}) = \prod_{r=1}^{R} q(\mathbf{W}_r), \quad \text{with} \quad q(\mathbf{W}_r) = \pi_w \mathcal{N}\left(\mathbf{M}_{w_r}, \sigma^2 \mathbf{I}_D\right) + (1-\pi_w)\mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}_D\right), \tag{3.14}$$

where $\pi_w \in [0,1]$, $\sigma^2 \approx 0$ and $\mathbf{M}_{w_r} \in \mathbb{R}^D$. This form of posterior leads to the sampling procedure which characterizes dropout Gal and Ghahramani (2016a,b). Given the choice of $\sigma^2 \approx 0$, $\mathbf{W}$ can be sampled by introducing Bernoulli variables

$$\mathbf{W} = \mathbf{M}_w \text{ Diag}[\mathbf{z}_w] \qquad \text{with} \qquad (\mathbf{z}_w)_i \sim \text{Bernoulli}(\pi_w), \tag{3.15}$$

and similarly for $\boldsymbol{\Psi}$ and $\boldsymbol{\Omega}$.

All variational parameters are optimized to maximize the lower bound of marginal likelihood which is defined as follows

$$\begin{aligned} \log\left[p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}\right] \geq & \text{E}_{q(\mathbf{W}, \boldsymbol{\Psi}, \boldsymbol{\Omega})}\left(\log\left[p\left(\mathbf{Y}|\mathbf{X}, \mathbf{W}, \boldsymbol{\Psi}, \boldsymbol{\Omega}, \boldsymbol{\theta}\right)\right]\right) \\ & - \text{KL}\left[q\left(\mathbf{W}, \boldsymbol{\Psi}, \boldsymbol{\Omega}\right) \| p\left(\mathbf{W}, \boldsymbol{\Psi}, \boldsymbol{\Omega}|\boldsymbol{\theta}\right)\right] \end{aligned} \tag{3.16}$$

The expectation in 3.16 can be unbiasedly estimated using Monte Carlo and also considering a mini-batch of size m

$$\begin{aligned}&\mathrm{E}_{q(\mathbf{W},\boldsymbol{\Psi},\boldsymbol{\Omega})}\left(\log\left[p\left(\mathbf{Y}|\mathbf{X},\mathbf{W},\boldsymbol{\Psi},\boldsymbol{\Omega},\boldsymbol{\theta}\right)\right]\right)\\&\approx\frac{N}{m}\frac{1}{N_{\mathrm{MC}}}\sum_{i=1}^{N_{\mathrm{MC}}}\sum_{k\in\mathcal{I}_m}\log\left[p\left(\mathbf{y}_k|\mathbf{x}_k,\mathbf{W}^{(i)},\boldsymbol{\Psi}^{(i)},\boldsymbol{\Omega}^{(i)},\boldsymbol{\theta}\right)\right],\end{aligned}\tag{3.17}$$

where $\mathbf{W}^{(i)},\boldsymbol{\Psi}^{(i)},\boldsymbol{\Omega}^{(i)}$ is a sample from $q(\mathbf{W},\boldsymbol{\Psi},\boldsymbol{\Omega})$, and can be obtained via 3.15. $\mathcal{I}_m$ is a set of m indices to select a mini-batch of training points. In classification, each individual $p\left(\mathbf{y}_k|\mathbf{x}_k,\mathbf{W}^{(i)},\boldsymbol{\Psi}^{(i)},\boldsymbol{\Omega}^{(i)},\boldsymbol{\theta}\right)$ can be computed using a softmax transformation. The KL term can be approximated following Gal and Ghahramani (2016a), noting that the fact that we are treating $\boldsymbol{\Omega}$ variationally, gives rise to extra terms that involve the GP length-scale ℓ:

$$\begin{aligned}&\mathrm{KL}\left[q\left(\mathbf{W},\boldsymbol{\Psi},\boldsymbol{\Omega}\right)\|p\left(\mathbf{W},\boldsymbol{\Psi},\boldsymbol{\Omega}|\boldsymbol{\theta}\right)\right]\\&\approx\frac{\pi_w}{2}\left\|\mathbf{M}_w\right\|^2+\frac{\pi_\psi}{2}\left\|\mathbf{M}_\psi\right\|^2+\frac{\ell^2\pi_\Omega}{2}\left\|\mathbf{M}_\Omega\right\|^2+N_{\mathrm{RF}}\,d\,\log\left(\ell^{-2}\right)\end{aligned}\tag{3.18}$$

3.6.2.2 CNN+GP(SORF)

In CNN+GP(SORF), our proposed variational inference scheme is similar to the one in CNN+GP(RF), except that $\boldsymbol{\Omega}$ is replaced by $l^{-1}\sqrt{N_{\mathrm{RF}}}\mathbf{H}\mathbf{D}_1\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_3$, with length-scale l and $\mathbf{D}_i=\mathrm{Diag}\left(\mathbf{d}_i\right)$ and $\mathbf{H}$ is the normalized Walsh-Hadamard matrix. Because $\mathbf{d}_i$ is Rademacher distributed, the form of prior and posterior in MCD proposed by Gal and Ghahramani (2016a,b) is inadequate. Therefore, we use the prior $p_\varepsilon\left(\mathbf{d}_i\right)=\mathcal{N}\left(\mathbf{d}_i|\mathbf{d}_i^*,\varepsilon^2\mathbf{I}_{N_{\mathrm{RF}}}\right)$ with $\mathbf{d}_i^*$ sampled from the Rademacher distribution and a small positive ε. The posterior $q\left(\mathbf{d}\right)$ is also composed by two Gaussian distribution as in CNN+GP(RF)

$$\begin{aligned}&q\left(\mathbf{d}_i\right)=\prod_{j=1}^{N_{\mathrm{RF}}}q\left(\left[\mathbf{d}_i\right]_j\right)\\&\text{, where }q\left(\left[\mathbf{d}_i\right]_j\right)=\pi_d\mathcal{N}\left(\mathbf{M}_{\left[\mathbf{d}_i\right]_j},\sigma^2\right)+\left(1-\pi_d\right)\mathcal{N}\left(\left[\mathbf{d}_i^*\right]_j,\sigma^2\right)\end{aligned}\tag{3.19}$$

with $\pi_d\in[0,1],\sigma^2\approx0$ and $\mathbf{M}_{\mathbf{d}_i}\in\mathbb{R}^{N_{\mathrm{RF}}}$. Following Gal and Ghahramani (2016a), we can approximate the KL term between $q\left(\mathbf{d}_i\right)$ and $p\left(\mathbf{d}_i\right)$

$$\mathrm{KL}\left(q\left(\mathbf{d}_i\right)\|p_\varepsilon\left(\mathbf{d}_i\right)\right)\approx\frac{\pi_d}{2\varepsilon^2}\|\mathbf{M}_{\mathbf{d}_i}-\mathbf{d}_i^*\|^2\tag{3.20}$$

In terms of implementation, we do not apply MCD to $\mathbf{d}_i-\mathbf{d}_i^*$ but on $\mathbf{d}_i$ directly. According to this choice, each element in $\mathbf{d}_i$ is sampled based on the variational

parameters $M_{\mathbf{d}_i - \mathbf{d}_i^*}$ as in 3.21. Thanks to this trick, the implementation of MCD scheme does not change for optimizing $\mathbf{d}_i$

$$\mathbf{d}_i = \begin{cases} \mathbf{M}_{\mathbf{d}_i - \mathbf{d}_i^*} + \mathbf{d}_i^*, & \text{with probability } \pi_d \\ \mathbf{d}_i^*, & \text{otherwise} \end{cases} \tag{3.21}$$

In figure 3.6, we report some experimental results to illustrate the impact of optimizing $\mathbf{d}_i$. For CIFAR10-LENET and CIFAR100-RESNET, the optimization of SORF parameters outperforms the case where spectral frequencies are fixed in terms of ERR, MNLL and BRIER. In the case of CIFAR10-RESNET, the gains are marginal.

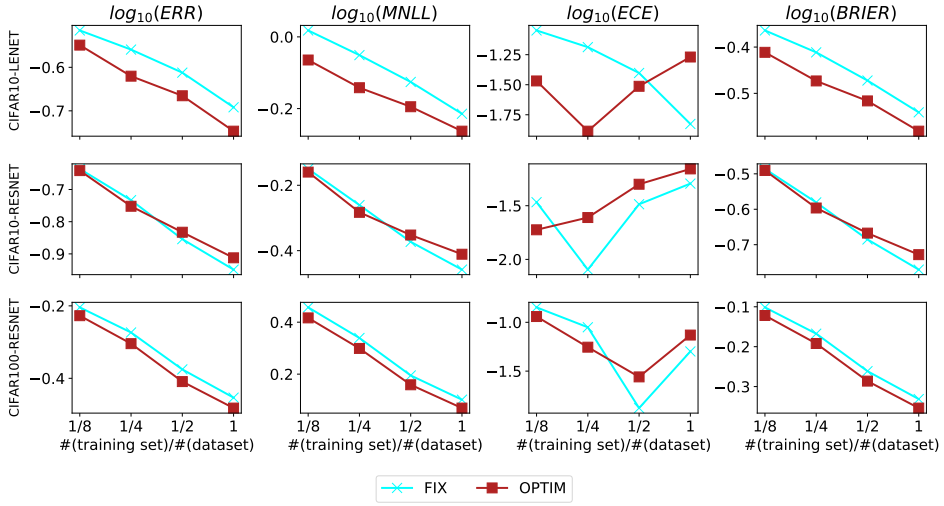


Figure 3.6 – Impact of optimization of SORF parameters

3.6.2.3 Optimization for covariance parameters

When using 3.18 to optimize all variational parameters pertaining to $q(\mathbf{W}, \mathbf{\Psi}, \mathbf{\Omega})$ jointly with covariance $\boldsymbol{\theta}$ we encountered some instabilities, and therefore we decided to report results when fixing the covariance parameters $\boldsymbol{\theta}$ in our paper. For the case where $\mathbf{\Omega}$ is not learned variationally we can simply draw $\mathbf{\Omega}$ from the prior $\mathcal{N}(\mathbf{\Omega}_{\cdot j}|\mathbf{0}, \Lambda^{-1})$ and consider the reparameterization:

$$\mathbf{\Omega}_{\cdot j} = \Lambda^{-\frac{1}{2}}\boldsymbol{\varepsilon}, \tag{3.22}$$

where $\varepsilon_i \sim \mathcal{N}(\varepsilon_i|0, 1)$ (Lázaro-Gredilla et al., 2010). This reparameterization allows for the update of covariance parameters $\boldsymbol{\theta}$ fixing the randomness in the sampling from $p(\mathbf{\Omega}|\boldsymbol{\theta})$. The results comparing CNN+GP(SORF) when updating or fixing $\boldsymbol{\theta}$ throughout optimization are reported in table 3.3. It is interesting to notice how fixing covariance parameters $\boldsymbol{\theta}$ leads to comparable performance to the case where they are learned.

Table 3.3 – Results on the proposed CNN+GP(SORF) when fixing or learning covariance parameters $\boldsymbol{\theta}$. All results were obtained on MNIST, CIFAR10, and CIFAR100 without subsampling the data. Please refer to table 1 in the main paper for details on the convolutional structure corresponding to SHALLOW and DEEP.

SHALLOW				
	MNIST		CIFAR10	
Metrics	Fixed	Learned	Fixed	Learned
ERR	0.006	0.005	0.203	0.192
MNLL	0.018	0.018	0.610	0.584
ECE	0.002	0.003	0.015	0.010
BRIER	0.009	0.008	0.288	0.271
DEEP				
	CIFAR10		CIFAR100	
Metrics	Fixed	Learned	Fixed	Learned
ERR	0.113	0.115	0.352	0.359
MNLL	0.348	0.355	1.264	1.287
ECE	0.051	0.054	0.050	0.054
BRIER	0.170	0.173	0.466	0.478

3.6.3 Variational inference of filters in GPDNN

In this section we report results when applying variational inference on the weights in GPDNN (Bradshaw et al., 2017). In order to do this, we implemented MCD for the convolutional parameters, similarly to what presented in the main paper for our CNN+GP(RF) model. The results in table 3.4 indicate that this improves the calibration and accuracy of GPDNN compared to optimizing the filters. In the case of a shallow convolutional architecture, the performance of CNN+GP(RF) and GPDNN are comparable, although in the deeper case CNN+GP(RF) achieves better performance. This supports the intuition that inferring convolutional parameters, ranther than optimizing them, leads to considerable improvements in calibration.

3.6.4 Reliability diagrams

In this section, we report the reliability diagram and histogram of predictive output for all methods with various datasets, i.e CIFAR10 and CIFAR100 and convolutional architectures, i.e LeNet and RESNET. We use the best configuration for CGP according to the implementation released by the Authors. In each figure, rows correspond with the dataset and convolutional architecture, while the column refer to the training size. After the training phase, all mod-

Table 3.4 – Results on the proposed CNN+GP(SORF) vs GPDNN when inferring convolutional parameters using MCD. All results were obtained on MNIST, CIFAR10, and CIFAR100 without subsampling the data. Please refer to table 1 in the main paper for details on the convolutional structure corresponding to SHALLOW and DEEP.

SHALLOW				
	MNIST		CIFAR10	
Metrics	CNN+GP(RF)	GPDNN	CNN+GP(RF)	GPDNN
ERR	0.005	0.005	0.172	0.172
MNLL	0.014	0.019	0.535	0.531
ECE	0.004	0.005	0.012	0.012
BRIER	0.0071	0.008	0.245	0.244
DEEP				
	CIFAR10		CIFAR100	
Metrics	CNN+GP(RF)	GPDNN	CNN+GP(RF)	GPDNN
ERR	0.111	0.190	0.351	0.820
MNLL	0.344	0.675	1.255	8.606
ECE	0.051	0.036	0.050	0.527
BRIER	0.168	0.278	0.466	1.268

els are evaluated on the entire testing set. The number of bins used to draw the reliability diagram is 20.

In each subfigure, the dashed line indicates perfect calibration. The horizontal axis is the softmax output ranging from 0 to 1. The vertical axis indicates accuracy rate for the red line or frequency for the green bars. The red dot is the real average accuracy at each bin, while the line segments at the red dots refer to the standard deviation of the accuracies. The green bar is the average frequency histogram at each bin of softmax values. The experiments of GPDNN, CGP, MCD-CIFAR10-LeNet and CNN+GP(RF) are repeated three times.

Having observed these figures, we see that regularizing convolutional filters has a huge impact on calibration. From figures 3.7, 3.8, 3.9 and 3.10 we see that CNNs and the previous combinations of GPs and CNNs are miscalibrated. From figure 3.12 and 3.13, instead, we see that Bayesian CNNs improve the reliability of the prediction, which is comparable with post-calibration.

It seems that there is a correlation between the histogram of predictive output and the reliability line. When the histogram is skewed to the right, the corresponding classifier is poorly calibrated.

Post calibration, MCD and CNN+GP(RF) (our method) are able to yeild calibrated classification.

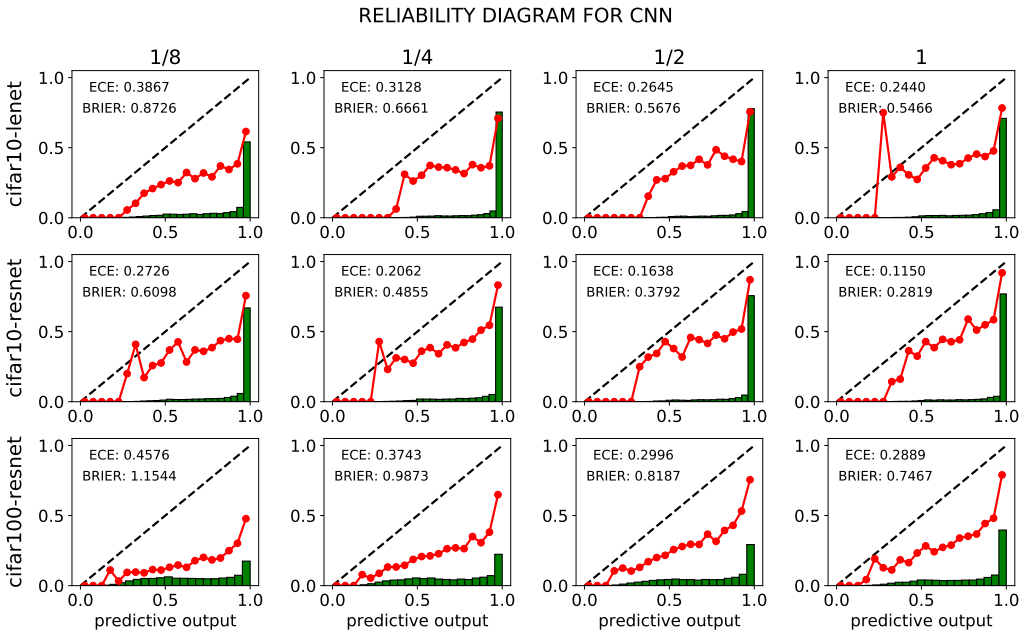


Figure 3.7 – Reliability diagrams for CNN

3.7 Conclusions

Despite the considerable interest in combining CNNs with GPs, little attention has been devoted to understand the implications in terms of the ability of these models to accurately quantify the level of uncertainty in predictions. This is the first work that highlights the issues of calibration of these models, showing that GPs cannot cure the issues of miscalibration in CNNs. We have proposed a novel combination of CNNs and GPs where the resulting model becomes a particular form of a Bayesian CNN for which inference using variational inference is straightforward. However, our results also indicate that combining CNNs and GPs does not generally improve the performance of standard CNNs. This can serve as a motivation for investigating new approximation methods for scalable inference in GP models and combinations with CNNs.

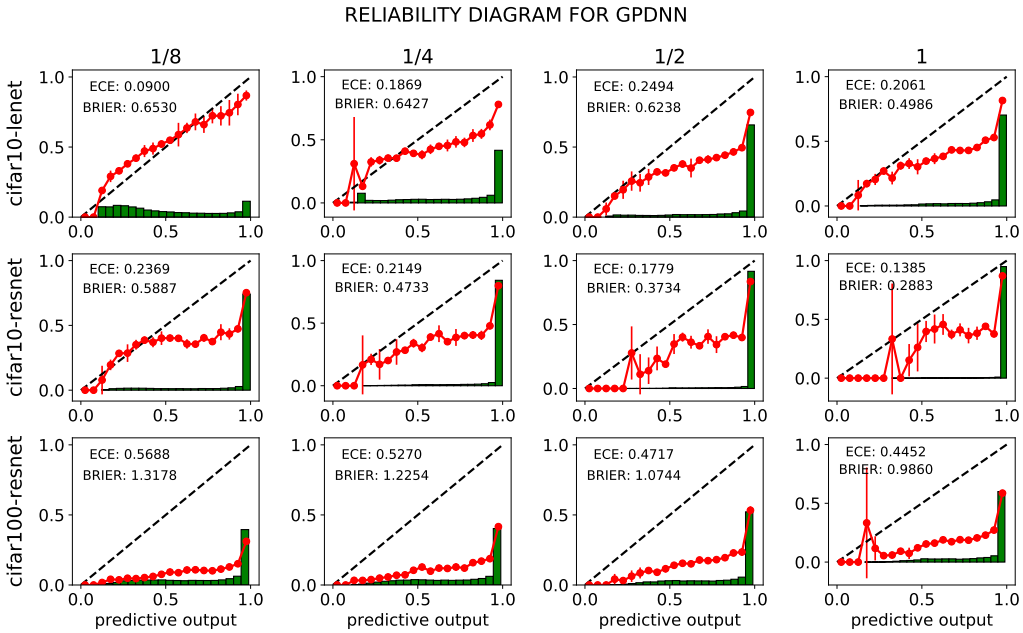


Figure 3.8 – Reliability diagrams for GPDNN

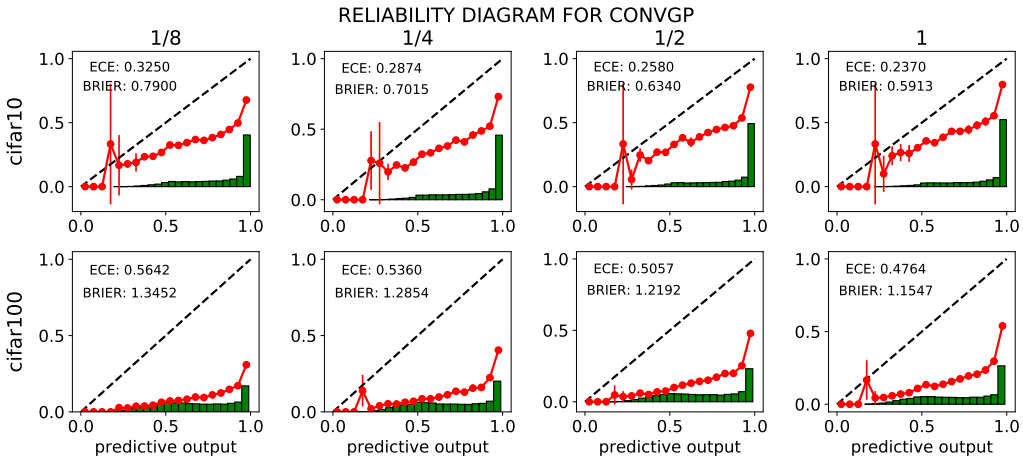


Figure 3.9 – Reliability diagrams for CGP

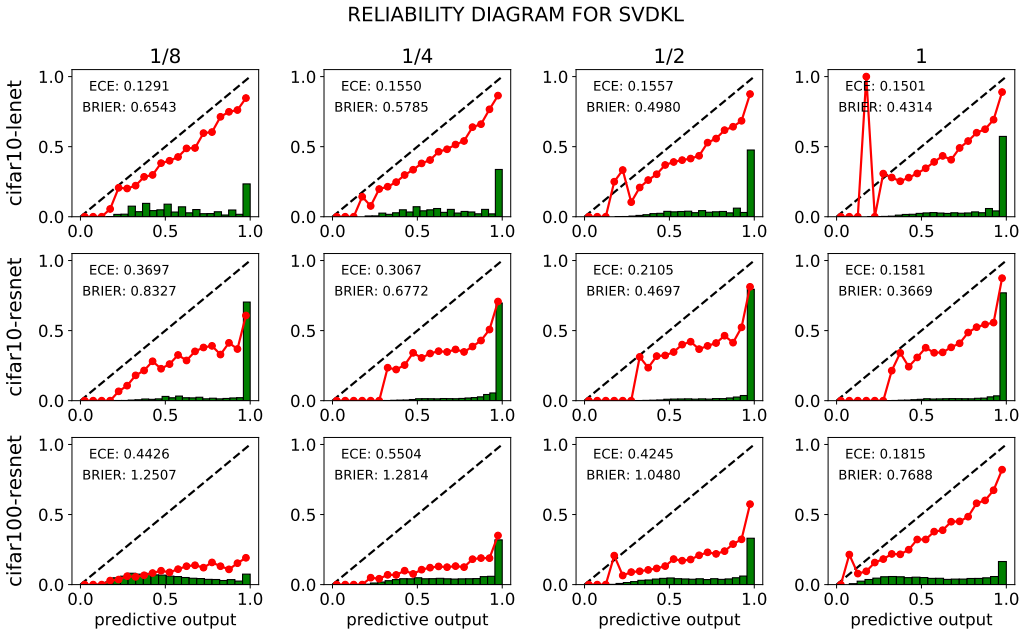


Figure 3.10 – Reliability diagrams for SVDKL

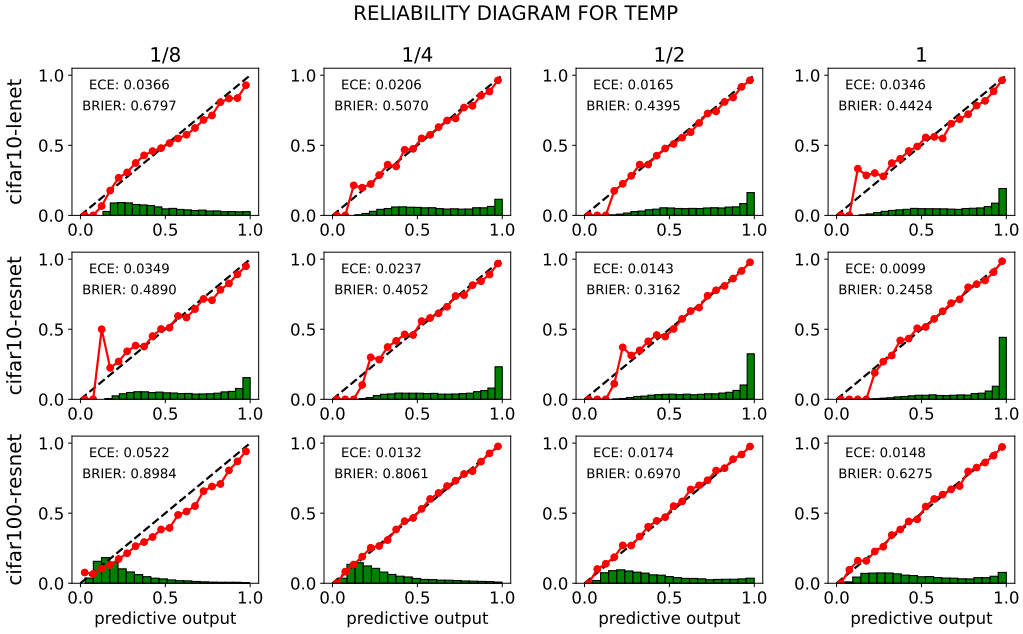


Figure 3.11 – Reliability diagrams for CNN+CAL

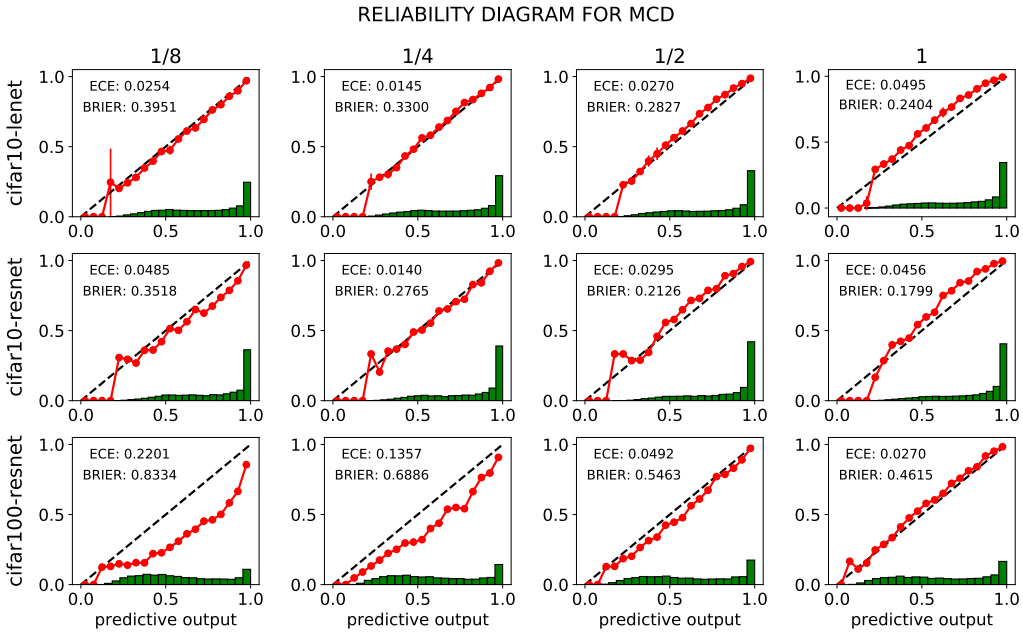


Figure 3.12 – Reliability diagrams for MCD

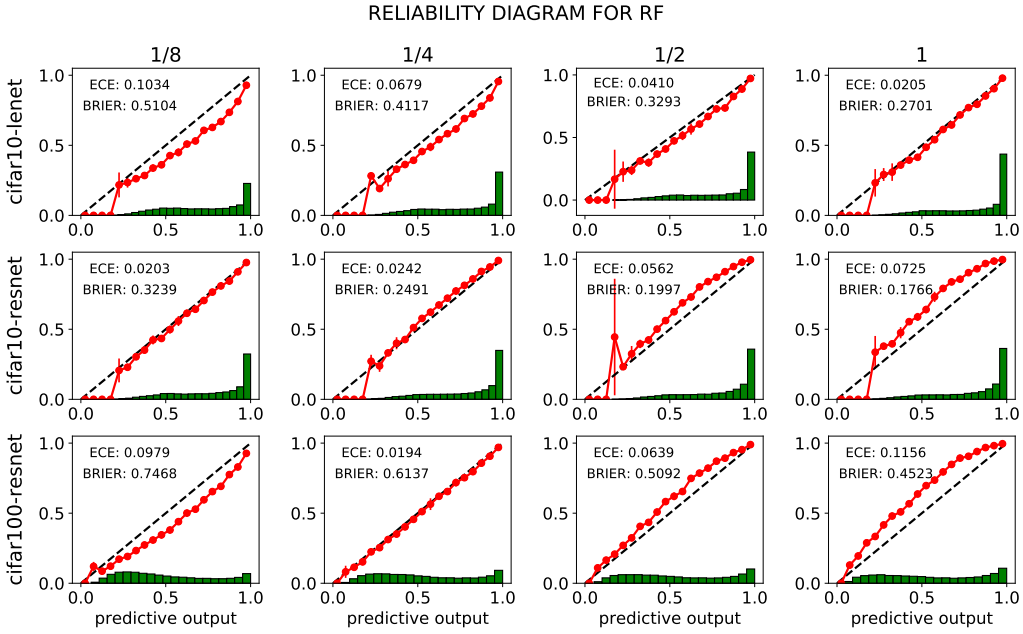


Figure 3.13 – Reliability diagrams for CNN+GP(RF)

4

Local and Global Approximation of Gaussian Processes

Approximations to Gaussian processes (GPs) based on inducing variables, combined with variational inference techniques, enable state-of-the-art sparse approaches to infer GPs at scale through mini-batch-based learning. In this work, we address one limitation of sparse GPs, which is due to the challenge in dealing with a large number of inducing variables without imposing a special structure on the inducing inputs. In particular, we introduce a novel hierarchical prior, which imposes sparsity on the set of inducing variables. We treat our model variationally, and we experimentally show considerable computational gains compared to standard sparse GPs when sparsity on the inducing variables is realized considering the nearest inducing inputs of a random mini-batch of the data. We perform an extensive experimental validation that demonstrates the effectiveness of our approach compared to the state-of-the-art. Our approach enables the possibility to use sparse GPs using a large number of inducing points without incurring a prohibitive computational cost.

4.1 Introduction

Gaussian Processes (GPs) (Rasmussen and Williams, 2006) offer a powerful framework to perform inference over functions; being Bayesian, GPs provide rigorous uncertainty quantification and prevent overfitting. However, the applicability of GPs on big datasets is hindered by their computational complexity of $\mathcal{O}(N^3)$, where N is the training size. This issue has fuelled a considerable amount of research towards scalable GP methodologies that operate on a set of *inducing variables* (Quiñonero Candela and Rasmussen, 2005). In the literature, there is a plethora of approaches that offer different treatments of the inducing variables (Lawrence et al., 2002; Seeger et al., 2003; Snelson and Ghahramani, 2005; Naish-Guzman and Holden, 2007; Titsias, 2009; Hensman

et al., 2013; Wilson and Nickisch, 2015; Hensman et al., 2015a). Some of the more recent approaches, such as Scalable Variational Gaussian Processes (SVGPs) (Hensman et al., 2015a), allow for the application of GPs to problems with millions of data points. In most applications of scalable GPs, these are approximated using M inducing points (IPs) , which results in a complexity of $\mathcal{O}(M^3)$. It has been shown recently by Burt et al. (2019) that it is possible to obtain an arbitrarily good approximation for a certain class of GP models (i.e. conjugate likelihoods, concentrated distribution for the training data) with M growing more slowly than N. However, the general case remains elusive and it is still possible that the required value for M may exceed a certain computational budget. Our result contributes to strengthen our belief that sparsity does not only enjoy desirable theoretical properties, but it also constitutes an extremely computationally efficient method in practice.

In this work, we push the limits of scalability and effectiveness of sparse GPs enabling a further reduction in complexity, which can be translated to higher accuracy by considering a larger set of inducing variables. The idea is to operate on a subset of H inducing points during training and prediction, with $H \ll M$, while maintaining a sparse approximation with M inducing variables. We formalize our strategy by imposing a sparsity-inducing structure on the prior over the inducing variables and by carrying out a variational formulation of this model. This extends the original SVGP framework and enables mini-batch-based optimization for the variational objective. We then consider ways to select the set of H inducing points based on neighbor information; at training time, for a given mini-batch, we activate H out of M inducing variables considering the nearest inducing inputs to the samples in the mini-batch, whereas at test time we select inducing variables corresponding to the inducing inputs which are nearest to the test data-points. We name our proposal *Sparse within a Sparse* GP (SWSGP). SWSGP is characterized by a number of attractive features: (i) it improves significantly the prediction quality using a small number of neighboring inducing inputs, and (ii) it accelerates the training phase, especially when the total number of inducing points becomes large. We extensively validate these properties on a variety of regression and classification tasks. We also showcase SWSGP on a large scale classification problem where we set $M = 100,000$; we are not aware of other approaches that can handle such a large set of inducing inputs without imposing some special structure on them (e.g., grid) or without considering one-dimensional inputs.

Hierarchical priors are often applied in Bayesian modeling to achieve compression and to improve flexibility (Molchanov et al., 2017; Louizos et al.,

2017). To the best of our knowledge, this work is the first to explore these ideas for the purposes of sparsifying the inducing set in sparse GPs.

4.2 Related work and background

Sparse GPs that operate on inducing inputs have been extensively studied in the last 20 years (Csató and Opper, 2002; Lawrence et al., 2002; Snelson and Ghahramani, 2005; Quiñonero Candela and Rasmussen, 2005; Naish-Guzman and Holden, 2007). Many attempts on sparse GPs specified inducing inputs by satisfying certain criteria that produce an informative set of inducing variables (Csató and Opper, 2002; Lawrence et al., 2002; Seeger et al., 2003). A different treatment has been proposed by Titsias (2009), which involves formulating the selection of inducing inputs as optimization of a variational lower bound to the marginal likelihood. The variational framework was later expanded so that stochastic optimization can be admitted, thus improving scalability for regression (Hensman et al., 2013) and classification (Hensman et al., 2015a). In a more recent work (Panos et al., 2018) scalability is addressed in terms of the dimensionality of the input. All the aforementioned methodologies share a computational complexity of $\mathcal{O}(M^3)$. Although there have been some attempts in the literature to infer the appropriate number of inducing points as well as the inducing inputs (Pourhabib et al., 2014a; Burt et al., 2019), a large number of inducing variables is desirable in improving the approximation to the posterior. In this work we present a methodology that builds on the SVGP framework (Hensman et al., 2015a) and reduces its complexity, thus increasing the potential of sparse GP application on even larger datasets and with a larger set of inducing variables.

A different approach to scalable GPs was introduced by Wilson and Nickisch (2015), namely Kernel Interpolation for Scalable Structured GPs (KISS-GP). This line of work involves arranging a large number of inducing inputs into a grid structure; this allows one to scale to very large datasets by means of fast linear algebra. The applicability of KISS-GP on higher-dimensional problems has been addressed by Wilson et al. (2015) by means of low-dimensional projections. A more recent extension allows for a constant-time variance prediction using Lanczos methods (Pleiss et al., 2018). Our work takes a different approach by keeping the GP prior intact, and by imposing sparsity on the set of inducing variables.

Local approximation of GPs inspired by the the concept of divide-and-conquer is also a practical solution to implement scalable GPs (Kim et al., 2005; Urta-

sun and Darrell, 2008; Datta et al., 2016; **?**; **?**) which allows GPs to work on large-scale datasets. In our work, we use neighbour information in a different way, by incorporating it in a certain hierarchical structure of the auxiliary variables through a variational scheme.

4.2.1 Scalable Variational Gaussian Processes

Consider a supervised learning problem with inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$ associated with labels $\mathbf{y} = (y_1, \ldots, y_N)^\top$. Given a set of latent variables $\mathbf{f} = (f_1, \ldots, f_N)^\top$, GP models assume that labels are stochastic realizations based on $\mathbf{f}$ and a likelihood function $p(\mathbf{y} \mid \mathbf{f})$. In SVGPs, the set of inducing points is characterized by inducing inputs $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_M)^\top$ and inducing variables $\mathbf{u} = (u_1, \ldots, u_M)^\top$. Regarding $\mathbf{f}$ and $\mathbf{u}$, we have the following joint prior:

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K_X} & \mathbf{K_{X,Z}} \\ \mathbf{K_{Z,X}} & \mathbf{K_Z} \end{bmatrix}\right), \tag{4.1}$$

where $\mathbf{K_X}$, $\mathbf{K_Z}$ and $\mathbf{K_{X,Z}}$ are covariance matrices evaluated at the inputs indicated by the subscripts. The posterior over inducing variables is approximated by a variational distribution $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m}, \mathbf{S})$, while keeping the exact conditional $p(\mathbf{f} \mid \mathbf{u})$ intact, that is $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})$. The variational parameters $\mathbf{m}$ and $\mathbf{S}$, as well as the inputs $\mathbf{Z}$, are optimized by maximizing a lower bound on the marginal likelihood $p(\mathbf{y} \mid \mathbf{X}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f} \mid \mathbf{X})d\mathbf{f}$. The lower bound on $\log p(\mathbf{y} \mid \mathbf{X})$ can be obtained by considering the form of $q(\mathbf{f}, \mathbf{u})$ above and by applying Jensen's inequality:

$$\mathrm{E}_{q(\mathbf{f})} \log p(\mathbf{y} \mid \mathbf{f}) - \mathrm{KL}(q(\mathbf{u}) \,\|\, p(\mathbf{u})). \tag{4.2}$$

The approximate posterior $q(\mathbf{f})$ can be computed by integrating out $\mathbf{u}$: $q(\mathbf{f}) = \int q(\mathbf{u})\, p(\mathbf{f} \,|\mathbf{u})\, d\mathbf{u}$. Thanks to the Gaussian form of $q(\mathbf{u})$, $q(\mathbf{f})$ can be computed analytically:

$$q(\mathbf{f}) = \mathcal{N}(\mathbf{f} \mid \mathbf{Am},\ \mathbf{K_X} + \mathbf{A}(\mathbf{S} - \mathbf{K_Z})\mathbf{A}), \tag{4.3}$$

where $\mathbf{A} = \mathbf{K_{X,Z}}\mathbf{K_Z^{-1}}$. When the likelihood factorizes over training points, the lower bound can be re-written as:

$$\textstyle\sum_{i=1}^{N} \mathrm{E}_{q(f_i)} [\log p(y_i \mid f_i)] - \mathrm{KL}(q(\mathbf{u}) \,\|\, p(\mathbf{u})). \tag{4.4}$$

Each term of the one-dimensional expectation of the log-likelihood can be computed by Gauss-Hermite quadrature for any likelihoods (and analytically for the Gaussian likelihood). The $\mathrm{KL}(q(\mathbf{u}) \,\|\, p(\mathbf{u}))$ term can be computed analytically given that $q(\mathbf{u})$ and $p(\mathbf{u})$ are both Gaussian. To maintain positive-definiteness of $\mathbf{S}$ and perform unconstrained optimization, $\mathbf{S}$ is parametrized as $\mathbf{S} = \mathbf{L}\mathbf{L}^T$, with $\mathbf{L}$ lower triangular.

4.3 Sparse within Sparse Gaussian Processes

We present a novel formulation of sparse GPs, which permits the use of a random subset of the inducing points with little loss in performance. We introduce a set of binary random variables $\mathbf{w} \in \{0,1\}^M$ to govern the inclusion of inducing inputs $\mathbf{Z}$ and the corresponding variables $\mathbf{u}$. We then employ these random variables to define a hierarchical structure on the prior as follows:

$$p\left(\mathbf{u} \mid \mathbf{w}\right) = \mathcal{N}\left(\mathbf{0}, \mathbf{D}_{\mathbf{w}}\mathbf{K}_{\mathbf{Z}}\mathbf{D}_{\mathbf{w}}\right), \tag{4.5}$$

where $\mathbf{D}_{\mathbf{w}} = \mathrm{Diag}\left(\mathbf{w}\right)$, and $\mathbf{w} \sim p\left(\mathbf{w}\right)$. Although the marginalized prior $p(\mathbf{u})$ is not Gaussian, it is possible to use the joint $p(\mathbf{u}, \mathbf{w}) = p(\mathbf{u} \mid \mathbf{w})\, p(\mathbf{w})$ within a variational scheme. We thus consider a random subset of the inducing points during the evaluation of the prior in the variational scheme that follows; no inducing points are permanently removed. Regarding $p(\mathbf{w})$, we consider an implicit distribution: its analytical form is unknown, but we can draw samples from it. Later, we will consider $p(\mathbf{w})$ based on the nearest inducing inputs to random mini-batches of data.

4.3.0.0.1 Remarks on the prior over f

Our strategy simply assumes a certain structure on the auxiliary variables, but it *has no effect* on the prior over $\mathbf{f}$; the latter remains unchanged. Let $\mathcal{I}$ and $\mathcal{J}$ bet the sets of indices such that $\mathbf{w}_{\mathcal{I}} = \mathbf{1}$ and $\mathbf{w}_{\mathcal{J}} = \mathbf{0}$. Given an appropriate ordering, the conditional $\mathbf{u} \mid \mathbf{w}$ is effectively the element-wise product $[\mathbf{u}_{\mathcal{I}}, \mathbf{u}_{\mathcal{J}}]^{\top} = \mathbf{u} \circ \mathbf{w}$. This reduces the variances and covariances of some elements of $\mathbf{u}$ to zero yielding a distribution of this form:

$$p\left(\mathbf{f}, \mathbf{u} \mid \mathbf{w}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{Z}_{\mathcal{I}}} & \mathbf{0}\\ \mathbf{K}_{\mathbf{Z}_{\mathcal{I}},\mathbf{X}} & \mathbf{K}_{\mathbf{Z}_{\mathcal{I}}} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}\right) \tag{4.6}$$

The rows and columns of $\mathbf{u}_{\mathcal{J}}$ can simply be ignored. Regardless of the value of $\mathbf{w}$, the conditional $\mathbf{f}, \mathbf{u}_{\mathcal{I}} \mid \mathbf{w}$ is always a Gaussian marginal, as it is a subset of Gaussian variables. The marginalized $p(\mathbf{f}, \mathbf{u}) = \int p(\mathbf{f}, \mathbf{u} \mid \mathbf{w})\, p(\mathbf{w}) d\mathbf{w}$ is mixture of Gaussian densities, where the marginal over $\mathbf{f}$ is the same for every component of the mixture.

The effect on $\mathbf{f}$ is demonstrated in Figure 4.1, where we sample from the (non-Gaussian) marginalized prior $p(\mathbf{u})$ in two steps: first we consider an arbitrary random subset $\mathbf{u}_{\mathcal{I}}$, and then we sample from $p(\mathbf{u}_{\mathcal{I}}) \equiv p(\mathbf{u} \mid \mathbf{w})$. Finally, $\mathbf{f}$ samples are drawn from $p(\mathbf{f} \mid \mathbf{u}_{\mathcal{I}})$, which only involves the selected inducing variables $\mathbf{u}_{\mathcal{I}}$. Following Eq. (4.1), the conditional $\mathbf{f} \mid \mathbf{u}$ is normally-distributed with mean $\mathbf{m}_{\mathbf{f}|\mathbf{u}_{\mathcal{I}}} = \mathbf{K}_{\mathbf{X},\mathbf{Z}_{\mathcal{I}}}\mathbf{K}_{\mathbf{Z}_{\mathcal{I}}}^{-1}\mathbf{u}_{\mathcal{I}}$ and covariance $\mathbf{S}_{\mathbf{f}|\mathbf{u}_{\mathcal{I}}} = \mathbf{K}_{\mathbf{X}} -$

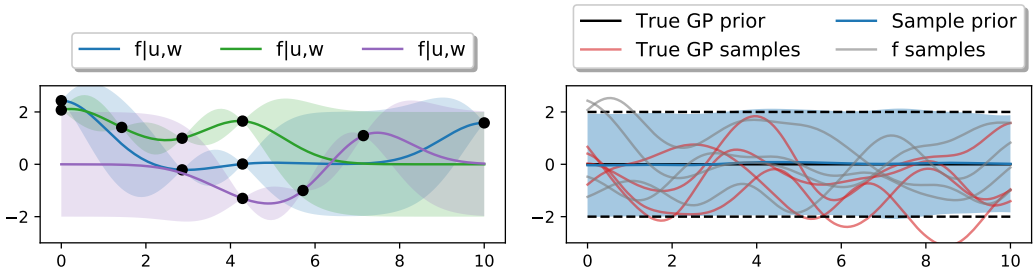


Figure 4.1 – The choice of inducing points does not affect the prior samples drawn from $p(\mathbf{f})$. Left: visualizations of $\mathbf{f} \mid \mathbf{u}, \mathbf{w}$ for different samples of $\mathbf{w}$. Right: comparison of the marginalised (w.r.t. $\mathbf{u}, \mathbf{w}$) prior over $\mathbf{f}$, against the true $p(\mathbf{f})$.

$\mathbf{K}_{\mathbf{X},\mathbf{Z}_\mathcal{I}} \mathbf{K}_{\mathbf{Z}_\mathcal{I}}^{-1} \mathbf{K}_{\mathbf{Z}_\mathcal{I},\mathbf{X}}$. These conditionals can be seen for different samples of $\mathbf{u}, \mathbf{w}$ in the left side of Figure 4.1, while in the right side we compare the marginalized prior over $\mathbf{f}$ against the true GP prior.

Of course, although the prior remains unchanged, that is not the case for the posterior approximation. It is well known that the choice of inducing inputs has an effect on the variational posterior (Titsias, 2009; Burt et al., 2019). Our choice to impose a hierarchical structure to the inducing variables through $\mathbf{w}$ effectively changes the model compared to SVGP, and we adapt the variational scheme accordingly.

4.3.1 Lower bound on marginal likelihood

By introducing $\mathbf{u}, \mathbf{w}$ and using Jensen's inequality, the lower bound on $\log p(\mathbf{y})$ can be obtained as follows

$$\mathrm{E}_{q(\mathbf{u},\mathbf{w})} \log p(\mathbf{y} \mid \mathbf{u}, \mathbf{w}) - \mathrm{KL}\left(q(\mathbf{u}, \mathbf{w}) \,\|p(\mathbf{u}, \mathbf{w})\right), \tag{4.7}$$

where we choose the variational distribution q to reflect the hierarchical structure of the prior, i.e. $q(\mathbf{u}, \mathbf{w}) = q(\mathbf{u} \mid \mathbf{w})\, p(\mathbf{w})$. This choice enforces sparsity over the approximate posterior q; the variational parameters are shared among the conditionals $q(\mathbf{u} \mid \mathbf{w})$, for which we assume:

$$q(\mathbf{u} \mid \mathbf{w}) = \mathcal{N}(\mathbf{u} \mid \mathbf{D}_\mathbf{w}\mathbf{m}, \mathbf{D}_\mathbf{w}\mathbf{S}\mathbf{D}_\mathbf{w}) \tag{4.8}$$

By maximizing the variational bounds that follow, we impose a q that performs well under a sparsified inducing set. We continue by applying Jensen's inequality on $p(\mathbf{y} \mid \mathbf{u}, \mathbf{w})$, obtaining:

$$\log p(\mathbf{y} \mid \mathbf{u}, \mathbf{w}) \geq \mathrm{E}_{p(\mathbf{f} \mid \mathbf{u},\mathbf{w})} \log p(\mathbf{y} \mid \mathbf{f}) \tag{4.9}$$

We can now substitute (4.9) into (4.7), obtaining a bound where we expand $q(\mathbf{u}, \mathbf{w})$ as $q(\mathbf{u}|\mathbf{w})\, p(\mathbf{w})$. By making this assumption, we obtain the following evidence lower bound $\mathcal{L}_{\text{ELBO}}$:

$$\sum_{n=1}^{N} \mathrm{E}_{p(\mathbf{w})} \Big[\mathrm{E}_{q(\mathbf{u}|\mathbf{w})} \mathrm{E}_{p(f_n|\mathbf{u},\mathbf{w})} \log p(y_n|f_n) - \frac{1}{N} \mathrm{KL}\left(q(\mathbf{u}|\mathbf{w}) \,\middle\|\, p(\mathbf{u}|\mathbf{w}) \right) \Big] \tag{4.10}$$

Recall that $p(\mathbf{w})$ is implicit: although we do not make any particular assumptions about its analytical form, we can draw samples from it. Using MC sampling from $p(\mathbf{w})$, we can obtain the approximation $\tilde{\mathcal{L}}_{\text{ELBO}}$:

$$\sum_{n=1}^{N} \Big[\mathrm{E}_{q(\mathbf{u}|\tilde{\mathbf{w}}^{(n)})} \mathrm{E}_{p(f_n|\mathbf{u},\tilde{\mathbf{w}}^{(n)})} \log p(y_n|f_n) - \frac{1}{N} \mathrm{KL}\left(q\left(\mathbf{u}|\tilde{\mathbf{w}}^{(n)}\right) \,\middle\|\, p\left(\mathbf{u}|\tilde{\mathbf{w}}^{(n)}\right) \right) \Big], \tag{4.11}$$

where $\tilde{\mathbf{w}}^{(n)}$ is sampled from $p(\mathbf{w})$.

4.3.1.0.1 Sampling from the set of inducing points.

Recall that any sample $\tilde{\mathbf{w}}$ from $p(\mathbf{w})$ is a binary vector, i.e. $\mathbf{w} \in \{0,1\}^M$. In case all elements of $\mathbf{w}$ are set to one, our approach recovers the original SVGP with computational cost of $\mathcal{O}(M^3)$ coming from computing $p(f_n|\mathbf{u}, \tilde{\mathbf{w}} = \mathbf{1})$ and $\mathrm{KL}(q(\mathbf{u}|\mathbf{w}) \| p(\mathbf{u}|\mathbf{w}))$ in the ELBO. When a $\tilde{w}_i$ is set to zero, the entries of the i-th row and i-th column of the covariance matrix in $p(\mathbf{u}|\mathbf{w})$ and $q(\mathbf{u}|\mathbf{w})$ are zero. This means that the i-th variable becomes unnecessary, so we get rid of i-th row and column in these matrices, and also eliminate the i-th element in mean vectors of $q(\mathbf{u}|\mathbf{w})$ and $p(\mathbf{u}|\mathbf{w})$. This is equivalent to selecting a set of active inducing points in each training iteration.

4.3.2 H-nearest inducing inputs

Despite the fact that $p(\mathbf{w})$ is an implicit distribution, we have been able to define and calculate a variational bound, assuming we can sample from $p(\mathbf{w})$. We shall now describe our sampling strategy, which relies on neighbor information of random mini-batches.

In order to explain the idea conveniently, we introduce $\mathbf{Z}_{\mathbf{x}}^{H}$ as the set of H-nearest inducing inputs. Intuitively, the prediction for an unseen data $\mathbf{x}$ using $\mathbf{Z}_{\mathbf{x}}^{H}$ is a good approximation of the prediction using all M inducing points,

that is $\mathbf{Z}_{\mathbf{x}}^{M}$. This can be verified by looking at the predictive mean, which is expressed as a linear combination of kernel functions evaluated between training points and a test point, as in Eq. (4.3). The majority of the contribution is given by the inducing points with the largest kernel values, so we can use this as a criterion to establish whether an inducing input is "close" to an input vector (the effect of different kernels on the definition of nearest neighbors is explored in the supplement). With this intuition, $p(\mathbf{w})$ becomes a deterministic function $w(\mathbf{x})$ indicating which inducing inputs are activated. For mini-batch-based training, the value of $\mathbf{w}$ remains random, as it depends on the elements $\mathbf{x}$ that are selected in the random mini-batch; this materializes the sampling from the implicit distribution $p(\mathbf{w})$. The maximization of the ELBO in the setting described is summarized in Algorithm 2 (SWSGP). At

Algorithm 2 Sparse within sparse GP (SWSGP).
Input: $\mathcal{D}$, H, M.
Result: The optimum of trainable parameters $\boldsymbol{\theta}$.

Initialize $\boldsymbol{\theta}$, i.e. kernel's parameters, $\mathbf{Z}$, $\mathbf{m}$ and $\mathbf{S}$.
while stopping criteria is False **do**
  ELL $\leftarrow 0$ and KL $\leftarrow 0$.
  Sample mini-batch $\mathcal{I}$ of size n from $\mathcal{D}$.
  for $(\mathbf{x}_i, y_i) \in \mathcal{I}$ **do**
    Find $\mathbf{Z}_{\mathbf{x}_i}^{H}$, i.e. the H-nearest $\mathbf{Z}$ to $\mathbf{x}_i$.
    Compute $w(\mathbf{x}_i)$ using $\mathbf{Z}_{\mathbf{x}_i}^{H}$ as in (4.12)
    Extract $\mathbf{m}_{w(\mathbf{x}_i)}$ and $\mathbf{S}_{w(\mathbf{x}_i)}$ from $\mathbf{m}$ and $\mathbf{S}$.
    Compute $q(f_i | w(\mathbf{x}_i))$ as in (4.13).
    ELL $\leftarrow$ ELL $+ \mathrm{E}_{q(f_i|w(\mathbf{x}_i))} \log p(y_i | f_i)$.
    KL $\leftarrow$ KL $+ \mathrm{KL}\left(q\left(\mathbf{u}_{w(\mathbf{x}_i)}\right) \| p\left(\mathbf{u}_{w(\mathbf{x}_i)}\right)\right)$
  end for
  $\tilde{\mathcal{L}}_{\mathrm{ELBO}} \leftarrow \frac{N}{n}\mathrm{ELL} - \frac{1}{n}\mathrm{KL}$.
  Update $\boldsymbol{\theta}$ using the derivative of $\tilde{\mathcal{L}}_{\mathrm{ELBO}}$.
end while

test time, however, the inputs of interest are not random; we need to describe the predictive distribution in terms of the deterministic function $w(\mathbf{x})$. In fact, if we would like to approximate the predictive distribution at $\mathbf{x}_n$ using H-nearest inducing inputs to $\mathbf{x}$, i.e. $\mathbf{Z}_{\mathbf{x}_n}^{H}$, then $w(\mathbf{x}) = \left[w_{\mathbf{x}}^{(1)} ... w_{\mathbf{x}}^{(M)}\right]^{T}$ where,

$$w_{\mathbf{x}}^{(m)} = \begin{cases} 1 & \text{if } \mathbf{z}_m \in \mathbf{Z}_{\mathbf{x}}^{H} \\ 0 & \text{else} \end{cases}, \text{ with } m = 1, ..., M \tag{4.12}$$

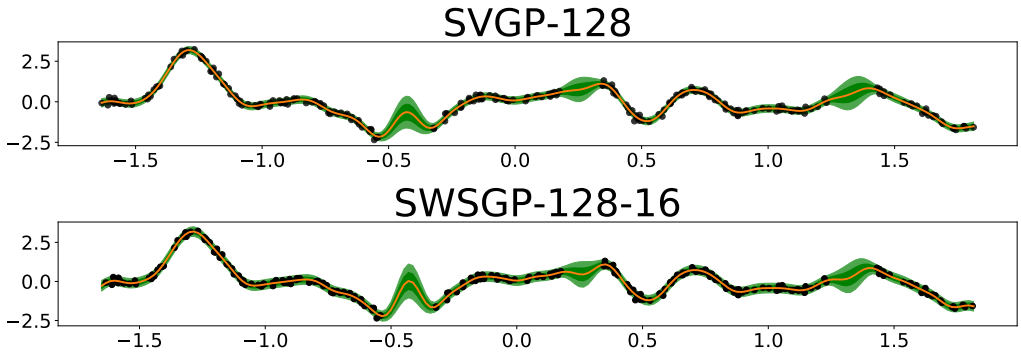


Figure 4.2 – Visualization of posterior distribution of SVGP and SWSGP. In both cases, we consider 128 inducing points; in terms of our scheme (SWSGP) we use 16 neighbors.

We extract the relevant elements using $w(\mathbf{x})$; for the mean, we have $\mathbf{m}_{w(x_i)} = \mathbf{D}_{w(x_i)}\mathbf{m}$, and for the covariance we select the appropriate rows and columns using $\mathbf{S}_{w(\mathbf{x}_i)} = \mathbf{D}_{w(\mathbf{x}_i)}\mathbf{S}\mathbf{D}_{w(\mathbf{x}_i)}$. The approximate posterior over f_i given $w(\mathbf{x}_i)$, i.e. $q(f_i \,|w(\mathbf{x}_i))$ is:

$$\mathcal{N}\Bigg(f_i \,|\mathbf{A}_{\mathbf{x}_i}\mathbf{m}_{w(\mathbf{x}_i)}, \\ \mathbf{K}_{\mathbf{x}_i} + \mathbf{A}_{\mathbf{x}_i}\left(\mathbf{S}_{w(\mathbf{x}_i)} - \mathbf{K}_{\mathbf{Z}^H_{\mathbf{x}_i}}\right)\mathbf{A}^\top_{\mathbf{x}_i}\Bigg), \tag{4.13}$$

where $\mathbf{A}_{\mathbf{x}_i} = \mathbf{K}_{\mathbf{x}_i,\mathbf{Z}^H_{\mathbf{x}_i}}\mathbf{K}^{-1}_{\mathbf{Z}^H_{\mathbf{x}_i}}$.

One-dimensional regression example. We visualize the posterior distribution for a synthetic dataset generated on a one-dimensional input space. We execute SVGP and SWSGP, and depict the posterior distributions of these two methods by showing the predictive means (orange lines) and the 95% credible intervals (shaded areas) in Figure 4.2. We consider identical settings for the two methods (i.e. 128 inducing points, kernel parameters, likelihood variance) and a neighbor area of 16 for SWSGP; a full account of the setup can be found in the supplement. We see that although the models are different, the predictive distributions appear remarkably similar. A more extensive evaluation follows in Section 4.4.

4.3.3 Complexity

The computational cost of SWSGP is dominated by lines 6, 8 and 9 in Algorithm 2. For each data point $(\mathbf{x}_i, y_i)$ in mini-batch $\mathcal{I}$, we need to find the H nearest inducing neighbors $\mathbf{Z}^H_{\mathbf{x}_i}$ for n points in line 6, where $n = |\mathcal{I}|$; this contributes to the worst-case complexity by $\mathcal{O}(nMH)$.

In line 8, we extract relevant parameters from $\mathbf{m}$ and $\mathbf{S}$. We focus on the cost of extracting $\mathbf{S}_{w(\mathbf{x}_i)}$ from $\mathbf{S}$. Similar to SVGP (Section 4.2.1), we consider $\mathbf{S} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is lower triangular. We extract $\mathbf{L}_{w(\mathbf{x}_i)} = \mathbf{D}_{w(\mathbf{x}_i)}\mathbf{L}$ which contains the rows of $\mathbf{L}$ that correspond to the Cholesky decomposition of $\mathbf{S}_{w(\mathbf{x}_i)} = \mathbf{L}_{w(\mathbf{x}_i)}\mathbf{L}_{w(\mathbf{x}_i)}^T$. The computational complexity of selecting the variational parameters is $\mathcal{O}(nMH^2)$.

Finally, the computation of approximating the predictive distribution in line 9 requires $\mathcal{O}(nH^3)$. The overall complexity for SWSGP in the general case is $\mathcal{O}(nMH + nMH^2 + nH^3)$, which is a significant improvement over the $\mathcal{O}(M^3)$ complexity of standard SVGP, assuming that $n, H \ll M$. If we choose $\mathbf{S}$ to be diagonal, the total complexity reduces to $\mathcal{O}(nMH + nH^3)$; if we additionally consider $\mathbf{Z}$ to be fixed, the computational cost is $O(nH^3)$. In the experiments of Section 4.4 we also explore these settings.

4.4 Experiments

In this section, we conduct experiments to evaluate SWSGP on a variety of experimental conditions. We denote our approach by SWSGP-M-H, where M inducing points are used and H determines how many neighbors are selected. We introduce SVGP-M, SVGP-H and SVGP-M-H as competitors; SVGP-M and SVGP-H are using M and H inducing points, respectively. SVGP-M-H, instead, refers to SVGP using M inducing points at training time and H-nearest inducing inputs at test time.

The comparison is carried out on some UCI data sets for regression and classification, i.e., POWERPLANT, KIN8NM, NAVAL, EEG, CREDIT, and SPAM. We also consider larger scale data sets, such as MNIST and the AIRLINE data. We use the Matérn-$5/2$ kernel in all cases except for the AIRLINE dataset, where the sum of a Matérn-$3/2$ and a linear kernel is used, similar to Hensman et al. (2015a). All models are trained using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 and a mini-batch size of 64. The likelihood for regression and binary classification are set to Gaussian and probit function, respectively. All models are trained over $100,000$ iterations except for the AIRLINE data set where models are trained for one million iterations. In regression tasks, we report the test root mean squared error (RMSE) and the test mean negative log-likelihood (MNLL), whereas we report the test error rate (ERR) and MNLL in classification tasks. The results are averaged over three folds.

4.4.1 Increasing the number of neighbors

We begin our experimental evaluation by investigating the behavior of SWSGP with respect to H. In Figure 4.3, we examine SWSGP on a two-dimensional

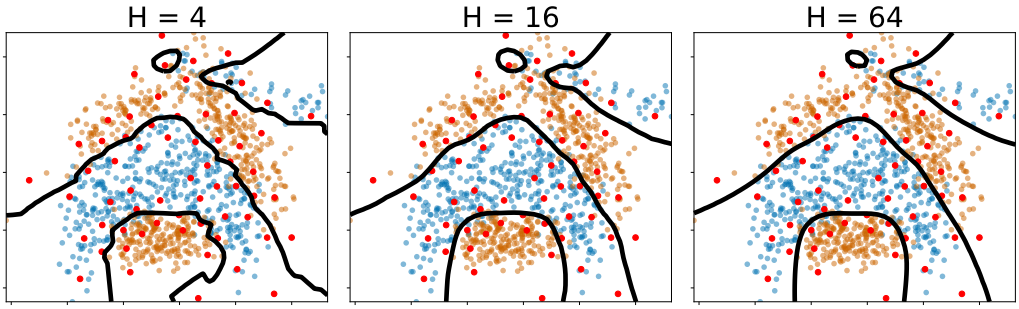


Figure 4.3 – Visualization of SWSGP on BANANA data sets with increasing H. The total number of inducing points M is fixed to 64, while the size of neighbor area H varies from 4 to 64. The red dots represent the inducing inputs. The orange and blue dots are training points from two different classes. The black lines are the contours of a classifier where the predictive mean is 0.5.

classification data set (BANANA), where M is fixed to 64 and H is increased from 4 to 64. In general, these boundaries remain sensible across the whole range of values of H, suggesting that SWSGP is able to work and converge well even though H is significantly less than M. We also observe that the contours of the classifier become smoother as H is increasing.

We then test SWSGP on other data sets with larger dimensional inputs. In these experiments, H is gradually increased to M. For POWERPLANT, KIN8NM, NAVAL, EEG, CREDIT and SPAM, M is set to 64, and for MNIST and AIRLINE, M is set to 512. In Fig. (4.4), we see that SWSGP-M-H consistently outperforms SVGP-M-H and SVGP-H. This suggests that including neighbor information at prediction time, combined with the use of a larger set of inducing points alone is not enough to obtain competitive performance, and that only thanks to the sparsity-inducing prior over latent variables, this yields improvements. Crucially, the performance obtained by SWSGP are comparable with those obtained by SVGP-M, while at each iteration only a subset of H out of M inducing points are updated, carrying a significant complextity reduction.

4.4.2 Increasing the number of inducing points

In this set of experiments, we show that the performance SWSGP improves when increasing the total number of inducing points, while keeping the number of active inducing points H fixed. We first illustrate this on the BANANA

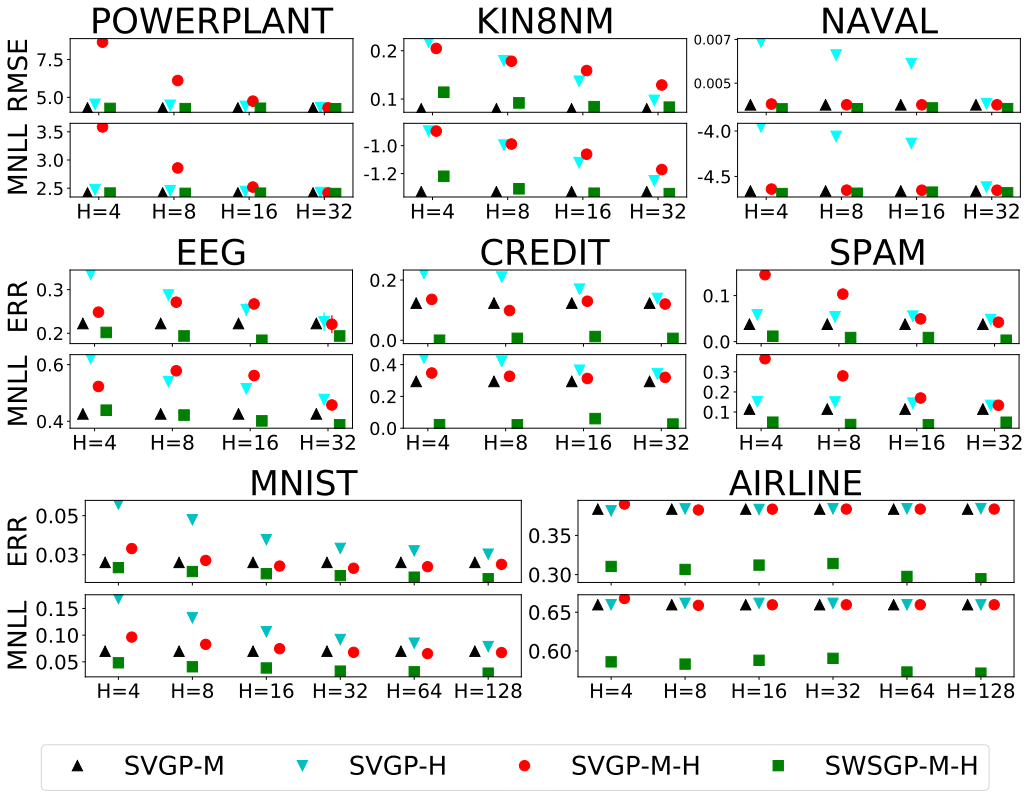


Figure 4.4 – Evaluation of SWSGP on high-dimensional data sets with increasing H. The black up-triangles are for SVGP with M inducing points, the cyan down-triangles are for SVGP with H inducing points, the red circles are for SVGP training with M inducing points and the prediction at an unseen data $\mathbf{x}$ are made by $\mathbf{Z}_{\mathbf{x}}^{H}$, and the green squares are for SWSGP. In these experiments, M is set to 64 and H varies from 4 to 32. Horizontal axis shows various configurations of H. The standard deviation of the error metrics over the different folds is represented by vertical bars; they are very small for most configurations.

data set, where H is fixed to 4 and M is gradually increased from 4 to 64. In Fig. (4.5) we see that the classification boundaries improve when increasing M.

We also investigate the impact of increasing H and M simultaneously. In each regression and classification data set, we test SWSGP with $H = 4, 8$ and $M = 8, 16, 32, 64$. The results shown in Fig. 4.6 indicate that using a small H is not detrimental to performance when M is large. In addition, SWSGP with a small H is comparable or better than SVGP in almost all cases.

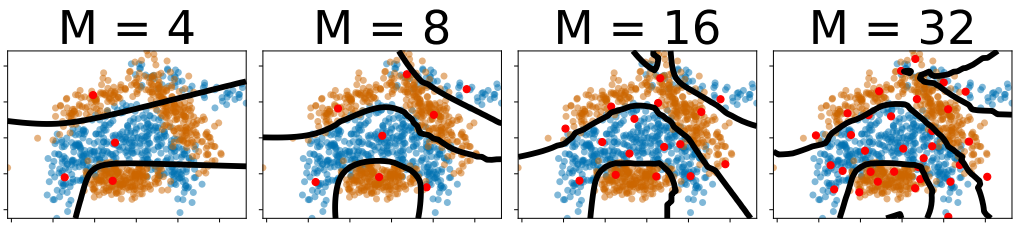
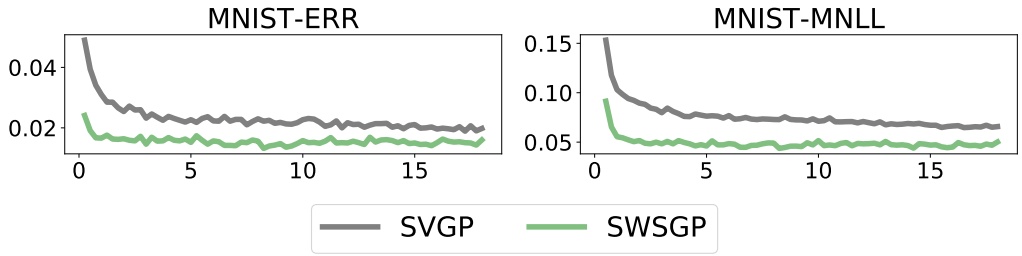


Figure 4.5 – Visualization of SWSGP on BANANA data sets with increasing M. The size of neighbor area H is set to 4. The total number of inducing points M varies from 4 to 64. The red dots represent inducing inputs. The orange and blue dots are the input points from the two different classes. The black lines are the contours of a classifier where the predictive mean is 0.5.

4.4.3 Running time

Table 4.1 – Comparison of running time between SVGP and SWSGP. In the table, each cell follows the format of [training time]|[testing time] (times are in milliseconds). In the figure, we show the progression of ERR (RMSE for regression case) and MNLL over training time. The black lines refer SVGP, and the green lines indicate SWSGP.

Configuration	POWERPLANT	EEG
SVGP-256	**22.83** | 2.89	**21.42** | 1.43
SWSGP-256-4	25.51 | **0.51**	26.18 | **0.56**
Configuration	**MNIST**	**AIRLINE**
SVGP-1024	516 | 21.6	465 | 45.8
SWSGP-1024-4	**233** | **1.77**	**157** | **0.78**



We show the training and testing times of SWSGP and SVGP in Tab. 4.1. In SVGP, we set $M = 256$ for POWERPLANT and KIN8NM, and 1024 for MNIST and AIRLINE, i.e. SVGP-256 and SVGP-1024. In our approach, we use the same M and we set H to 4 and M, i.e. SWSGP-256-4 and SWSGP-1024-4. Each cell of Tab. 4.1 follows the format of $t_1 \mid t_2$ where t_1 and t_2 indicate execution time of training and testing in milliseconds. The time t_1 is the averaged training time of a training iteration. The time t_2 is the averaged execution time to evaluate the predictive distribution on a test point. We stress that t_1 and t_2

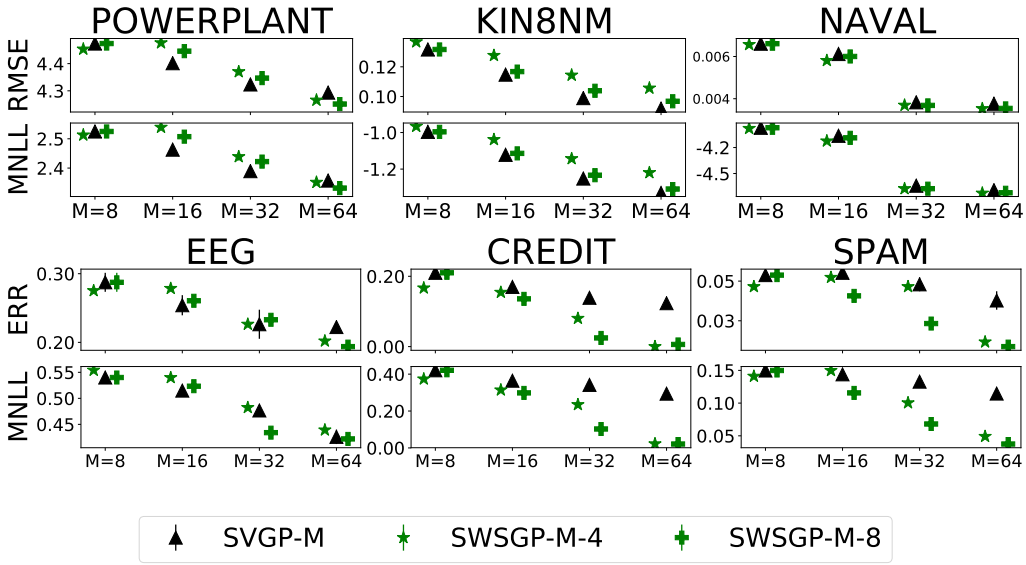


Figure 4.6 – Evaluation of SWSGP on high-dimensional data sets with increasing M. The black up-triangles are for SVGP with M inducing points. The green stars and plus are for SWSGP with H of 4 and 8 respectively. In these experiments, M varies from 4 to 64, as shown on horizontal axes. The standard deviation of the error metrics over the different folds is represented by vertical bars; they are very small for most configurations.

in SWSGP take into account the computation of finding neighbors inducing inputs for each data point. In SVGP, we assume that $\mathbf{K}_{\mathbf{Z}}^{-1}$ is pre-computed and saved after the training phase. Therefore, the computational cost to evaluate the predictive distribution on a single test point is $\mathcal{O}(M^2)$. The time t_2 in SVGP refers to the execution time of carrying out predictions with the complexity of $\mathcal{O}(M^2)$.

The results in Tab 4.1 show a consistent improvement at test time compared to SVGP across all values of H and M. At training time, the results show a trend dependent on the number M of inducing points. Not surprisingly, SWSGP offers limited improvements when M is small. Considering POWERPLANT and KIN8NM in which M is set to 256, SVGP is faster than SWSGP in terms of training time. This is because the inversion of a 256×256 matrix requires less time than finding the neighbors and inverting several 4×4 matrices. However, Tab 4.1 shows dramatic speedups compared to SVGP when the number of inducing points M is large. When $M = 1024$ on MNIST and AIRLINE, SWSGP-1024-4 is faster than SVGP-1024 in training time. This is due to the inversion of the 1024×1024 kernel matrix being a burden for SVGP, whereas SWSGP deals with much cheaper computations. Finally, we show the

progression of ERR and MNLL over training time when we train SVGP-1024 and SWSGP-1024-4 on MNIST. It becomes apparent that for large datasets our method achieves high levels of accuracy significantly more quickly in terms of running time compared to the standard SVGP.

4.4.4 Large-scale problems with a huge number of IPs

We showcase a large-scale classfication problem, where we illustrate that SWSGP enables the possibility to use sparse GPs with a massive number of inducing points without incurring a prohibitive computational cost. We employ the AIRLINE data set, featuring 5 million training points. We test SWSGP with $M = 100,000$ inducing points. We attempted to run SVGP with such a large M without success (out of memory in a system with 32GB of RAM). Therefore, as a baseline we report the results of SVGP with the configuration in Hensman et al. (2015a).

In SWSGP, we impose a diagonal matrix $\mathbf{S}$ in the variational distribution $q(\mathbf{u} \mid \mathbf{w})$, and we fix the position of the inducing inputs during training. By fixing the inducing inputs, we can operate with pre-computed information about which inducing inputs are neighbors of training inputs. Thanks to these settings, SWSGP's training phase requires $\mathcal{O}(nH^3)$ operations only, where n is the mini-batch size. Due to the appropriate choice of H and n, and the computational cost being independent of M, unlike SVGP, we can successfully run SWSGP with $M = 100,000$.

By setting H and the mini-batch size n to 100 and 16 respectively, in about 24 hours of training we could run SWSGP-100,000-100 for one million iterations. The ERR and MNLL of SWSGP-100,000-100 evaluated on the test set are 21% and 0.48, respectively, while the ERR and MNLL of SVGP-200 published in Hensman et al. (2015a) are about 34% and 0.61, respectively. To the best of our knowledge, SWSGP is the first to enable sparse GPs with such a large set of inducing points without imposing a grid structure on the inducing inputs. We conclude by reporting comparisons with other GP-based models. In particular, we compare against the Stochastic Variational Deep Kernel Learning (SVDKL) (Wilson et al., 2016) and the Deep GP approximated with random features (Cutajar et al., 2017). In the former, KISS-GP is trained on top of a deep neural network which is optimized during training, and in the latter the layers of a deep GP are approximated as parametric models using random feature expansions. Both competitors feature mini-batch-based learning, so this represents a challenging test for SWSGP. The results in Tab. 4.2 show that SWSGP is comparable with these competitors. We believe that this is

a remarkable result obtained by our shallow SWSGP, supporting the conclusions of previous works showing that advances in kernel methods can result in performance which are competitive with deep learning approaches (see, e.g., Rudi et al. (2017)).

Method	Data set	RMSE	MNLL
SWSGP-64-4	POWERPLANT	**4.29**	**2.42**
KISS-GP	POWERPLANT	11.26	5.78
SWSGP-100k-100	AIRLINE	0.21	0.48
SVDKL	AIRLINE	0.22	0.46
Deep GP random features	AIRLINE	0.21	0.46

Table 4.2 – Comparison of SWSGP, KISS-GP (Wilson and Nickisch, 2015), SVDKL (Wilson et al., 2016) and Deep GPs random features (Cutajar et al., 2017)

4.4.5 Comparison to Local GPs

We finally demonstrate that SWSGP behaves differently from other approaches that use local approximations of GPs. We consider two well-established approaches of local GPs proposed by Kim et al. (2005) and Urtasun and Darrell (2008). Following Liu et al. (2018b), we shall refer to these methods as *Inductive* GPs and *Transductive* GPs, respectively. We run all methods on two regression data sets: POWERPLANT and KIN8NM. We set the number of local experts to 64, and we use the same number of inducing points for SWSGP (with H either 4 or 8). As the size of POWERPLANT and KIN8NM are approximately 7000, we set the number of training points governed by a local expert to 100. For the local GP approaches, we choose 64 locations in the input space using the K-means algorithm, and for each location we choose 100 neighboring points; we then train the corresponding local GP expert. Regarding the testing phase, inductive GPs simply rely on the nearest local experts to an unseen point $\mathbf{x}_*$. Whereas for transductive GPs, we use 100 neighbors of $\mathbf{x}_*$ and the nearest local expert to make predictions. In table 4.3, we summarize RMSE and MNLL for all methods; SWSGP clearly outperforms the local GP approaches in terms of MNLL.

Method	POWERPLANT RMSE | MNLL	KIN8NM RMSE | MNLL
SWSGP-64-4	4.27 | 2.41	0.11 | −1.27
SWSGP-64-8	**4.24** | **2.40**	0.10 | **-1.38**
Inductive GPs	9.93 | 38.38	0.13 | −0.40
Transductive GPs	6.17 | 18.78	**0.09** | −0.65

Table 4.3 – Comparison with Local GP approximations.

4.5 Other results

4.5.1 Various options for H-nearest inducing points selection

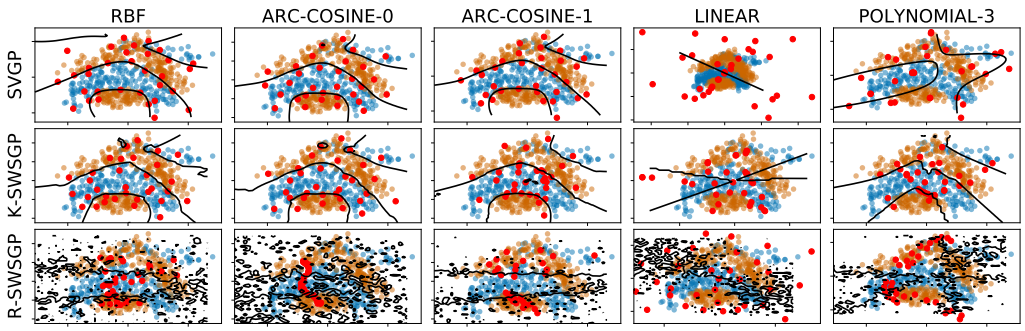


Figure 4.7 – SWSGP on various kernels and strategies for selecting the H-nearest inducing points.

As we discuss in the paper, the selection of H-nearest inducing points $\mathbf{Z}_x^H$ is made by using the kernel as a proxy to the concept of distance. Intuitively, a kernel defines the similarity between two points in the input space, which is more formally expressed as correlation. The kernel implicitly defines a kind of distance that we use to determine the active neiborhood. Thus, the selected neiborhood is dominated by the inducing points with largest kernel values.

In the main paper, we have used different versions of the Matérn kernel. We shall now explore the effect of our neiborhood-selection strategy on a number of different kernels, both stationary and non-stationary. We apply SWSGP on the BANANA data-set using different heuristics for the H-nearest inducing points selection. Let K-SWSGP denote what is essentially the vanilla version of our method, where the kernel-based heuristic is used as a proxy to distance. In the case of the RBF kernel, K-SWSGP essentially corresponds to the

Euclidean distance. We also examine a random-based heuristic (R-SWSGP) in which H-nearest inducing points are randomly chosen. In all cases, we set M and H as 32 and 8 respectively. We also compare against SVGP with M of 32.

In Fig. (4.7), we visualize the contours of classifiers of SVGP and SWSGP with various configurations. Clearly, R-SWSGP does not work, i.e. the contours are discontinuous and the locations of contours does not makes sense. Regarding the kernels RBF, ARC-COSINE-0 and ARC-COSINE-1, our method (K-SWSGP) seems to be virtually identical to SVGP. The advantages of K-SWSGP over SVGP are shown when using POLYNOMIAL-3. It is highly possible that the flexibility of variational distribution over inducing variables, i.e. $q(\mathbf{u})$, in SWSGP is the main reason for this difference.

4.5.2 Further visualizations on 1D examples

We demonstrate SWSGP on one-dimensional regression problem. We have generated a synthetic data-set by sampling inputs x_i from the interval $[-2, 2]$; the targets have been computed as $y_i = \sin(12x_i) + 0.66\cos(25x_i) + \varepsilon$, where ε is additive Gaussian noise with variance 0.1. Figure 4.8 summarizes the regression result for a fixed M, while the value of H varies from 4 to 64. We notice that the predictive means are nearly identical across the different sub-figures. These observations suggest that SWSGP is able to work and converge well even though H is significantly less than M.

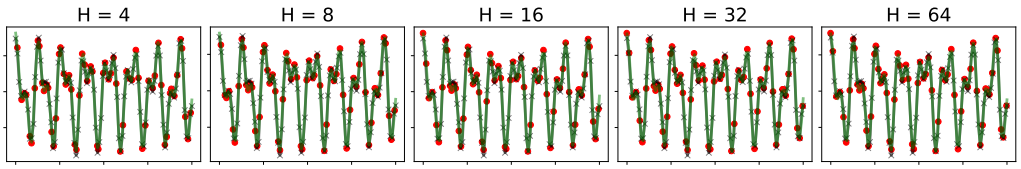


Figure 4.8 – SWSGP is applied on a one-dimensional data set, where M is fixed to 64 and H is increased gradually from 4 to 64. The red dots are inducing positions; the black crosses are testing points; the green line refers to predictive means.

We also show that the performance of SWSGP improves when increasing the total number of inducing points while keeping the number of active inducing points H fixed. We intuitively expect that a larger the total number of inducing points should translate to a more accurate model. In these experiments, the size of neighbor area is fixed to 4, i.e. $H = 4$, and the total number of inducing points are varies from 4 to 64. We see that the sequence of the predictive means in Fig. 4.9 are more and more accurate from left to right.

Although we are using a small neighbor area, our model is improved when increasing the total number of inducing points.

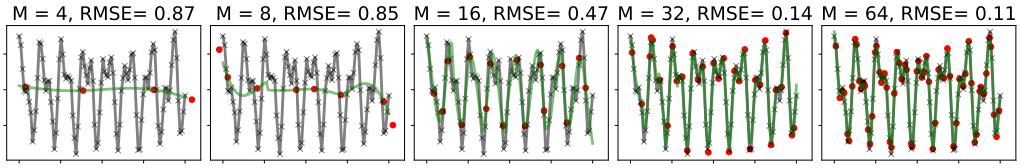


Figure 4.9 – SWSGP is applied on 1D. The red dots are inducing positions. The black crosses are testing samples. The green lines are predictive means. The title of each sub-figures shows M and corresponding RMSE.

4.6 Conclusions

Sparse approaches that rely on inducing points have met with success in reducing the complexity of GP regression and classification. However, these methods are limited by the number of inducing inputs that is required to obtain an accurate approximation of the true GP model. A large number of inducing inputs is often necessary in cases of very large datasets, which marks the limits of practical applications for most GP-based approaches.

In this work, we further improve the computational gains of sparse GPs. We proposed SWSGP, a novel methodology that imposes a hierarchical and sparsity-inducing effect on the prior over the inducing variables. This has been realized as a conditional GP given a random subset of the inducing points, which is defined as the nearest neighbors of random mini-batches of data. We have developed an appropriate variational bound which can be estimated in an unbiased way by means of mini-batches. We have performed an extensive experimental campaign that demonstrated the superior scalability properties of SWSGP compared to the state-of-the-art.

5 Conclusion

The models and techniques presented in this thesis are unified by the overarching goal of improving the calibration and scalability of Gaussian Processes. We conclude this thesis by summarizing the principal themes and contributions presented in the preceding chapters, with particular emphasis on their significance in the context of complementary work in this direction of research. This is followed by a brief outlook on possible avenues for future work where we indicate how one might go about achieving these objectives.

5.1 Themes and Contributions

In this thesis, we primarily investigated the following themes in relation to Gaussian processes:

• **Well-calibrated deep convolutional probabilistic model**. Developing models which are able to provide accurate predictions and reliable uncertainties has been a long-standing research topic attracting significant attention from machine learning community. Deep CNNs that have accomplished state-of-the-art results in a range of tasks have been illustrated to be miscalibrated, the depth of architecture are the main factor affecting calibration (Guo et al., 2017). Thinking of Bayesian priors as a form of regularization, it is natural to assume that Bayesian CNNs are an appropriate treatment for the problem of miscalibration of modern CNNs. Independently of the works on Bayesian CNNs implemented by Monte Carlo Dropout (Gal and Ghahramani, 2016b), there have been other attempts to give a probabilistic flavor to CNNs by combining them with Gaussian processes (Wilson et al., 2016; Bradshaw et al., 2017; van der Wilk et al., 2017). To the best of our knowledge, prior to our work there were no studies showing calibration properties of these Bayesian CNNs approaches. Hence, in Chapter 3, we investigated the calibration properties of Bayesian treatment on CNNs. Perhaps surprisingly, the results indicated that current combinations of CNNs and GPs are miscalibrated, with a tendency of

being overconfident in predictions. Consequently, by extending the random feature expansion approximation for DGPs (Cutajar et al., 2017), we proposed a novel combination of CNNs and GPs which is well-calibrated, and we validated it through several experimental results on image classification tasks. Furthermore, our model was extended by replacing the last fully-connected layers of CNNs with Deep GPs (Cutajar et al., 2017) and by employing structured random features to obtain faster and more compact GP approximations (Le et al., 2013; Yu et al., 2016).

• **Combination of global and local approximation.** Gaussian Processes Rasmussen and Williams (2006) offer a powerful statistical framework for inference on functions. However, the applicability of GPs on big datasets is hindered by the prohibitive complexity depending on training size N. Due to the rigorous uncertainty quantification of GPs, the inducing point-based sparse approximation of GPs have been extensively studied (Snelson and Ghahramani, 2005; Quiñonero Candela and Rasmussen, 2005; Titsias, 2009; Hensman et al., 2015a). The state-of-the-art approaches, e.g Scalable Variational Gaussian Processes (Hensman et al., 2015a), allows for the application of GPs to large-scale problems with a small number of inducing points M. As shown recently by Burt et al. (2019), it is possible to obtain an arbitrarily good approximation for a certain class of GP models with M growing more slowly than N. However, in general, it is still possible that the required value for M may exceed a certain computational budget. In Chapter 4, by imposing a sparsity-inducing structure on the prior over the inducing variables and by carrying out a variational formulation of this model, we pushed the limits of scalability and effectiveness of sparse GPs enabling a further reduction of computational complexity. Our experimental results showed that the use of unprecedented number of inducing points led to higher accuracy on AIRLINE which is a dataset with millions data points. In addition, we showed that our proposed model is able to know what it does not know by yielding sensible predictive uncertainties.

5.2 Future work

Beyond the discussion featured in this thesis, the themes explored in this body of work not only motivate immediate extensions for improvements, but also set the foundations for broader long-term objectives. In this section, we expand upon the directions for future work which we believe to be particularly pertinent to ongoing developments in both the theoretical and practical aspects of machine learning using GPs. We partition this discussion into the

overarching themes of (i) studying calibration properties of GPs regression; and (ii) proposing more elegant mixtures of CNNs and GPs; and (iii) adapting the state-of-the-art scalable GPs to online machine learning.

5.2.1 Calibrated GP regression

In addition to improving the scalability of GPs, producing reliable predictive uncertainties is also a primary goal for the application of GPs in the era of big data, especially when GPs are components of larger decision-making pipelines. This aspect can be evaluated by analyzing calibration properties mentioned in Chapter 3. While the reliability of the predictive uncertainties of Bayesian CNNs on classification tasks has been analyzed Guo et al. (2017); Lakshminarayanan et al. (2017); Tran et al. (2019), the calibration of GP-based regression methods has not been considered carefully. As mentioned in Kuleshov et al. (2018), the calibration property of regressors is evaluated by their predictive interval. A regressor is stated to be calibrated if p-percent credible intervals contain the true outcomes p-percent of the time. Starting with the novel vision about reliable regressors, investigating calibration properties of GPs on regression promises to be interesting. Some potential candidates reinforcing the model's calibration may be inspired by the preceding works, for example post-calibration by Platt scaling (Platt, 1999; Guo et al., 2017) or training with adversarial samples (Lakshminarayanan et al., 2017).

5.2.2 Elegant mixtures of CNNs and GPs

While studying on Bayesian CNN, we have realized that combining CNNs and GPs does not generally improve the performance of standard GPs. We speculate that the kernel's parameterization with a high number of parameter increases the risk of overfitting, and leads to overconfident tendency in predictions. As shown in Chapter 3, the Bayesian treatment on convolutional parameters enhances not only model's generalization but also model's calibration. However, the improvements of our approach carries a great computational cost due to repeated feed-forward procedure. This limit can serve as a motivation for investigating new approximation methods for scalable inference in GP models and combinations with CNNs.

Generally, the Bayesian flavor in the mixtures of CNNs and GPs can be strengthened by applying a full Bayesian treatment. For example, following our works in Chapter 3, the proposed models can be further improved by applying a Bayesian treatment on priors of parameters, which would result in the optimization of dropout rates of convolutional hyperparameters (Kingma et al.,

2015; Molchanov et al., 2017; Louizos et al., 2017). Along a similar vein, independent works replacing the fully-connected layers of CNNs by GPs (Wilson et al., 2016; Bradshaw et al., 2017; Tran et al., 2019), while Deep Convolutional Gaussian Processes (DCGPs) proposed by Blomqvist et al. (2018) substitutes GPs for convolutional filters. Another interesting approach could be applying the Bayesian formulation mentioned in Titsias and Lazaro-Gredilla (2013) together with sparsity inducing priors (Louizos et al., 2017; Molchanov et al., 2017) on DCGPs, a procedure of learning architecture is proposed, which not only accelerates computations but also allows one to approximately integrate out kernel hyperparameters, such as length-scales.

5.2.3 Adaptability to online machine learning

According to the extensive literature review in Liu et al. (2018b), local approximations are common approaches to implement scalable statistical inference systems. The uses of local approximations require to define the localization of experts, which directly affects to the assignments of data points to local experts. Likewise, in chapter 4, our proposal named Sparse-within-sparse Gaussian Processes (SWSGP) perceived as a combination of global and local approximations also relies on the way to select active inducing points for each inputs. On offline tasks, SWSGP was shown to be effective in terms of accuracy and complexity. With the application of online machine learning wherein training sets are constantly evolving, the selection of active inducing points based upon spatial or temporal distance, which is implemented in SWSGP, may ignore the information related to periodic patterns. In such scenario, a kernel-based distance seems to be more appropriate because the kernel intuitively determines the correlation between two points in the input space. More general, by perceiving the selection of active inducing points as a gating function, the input-dependent Dirichlet Process (Rasmussen and Ghahramani, 2002) and Polya urn distribution (Meeds and Osindero, 2006) can automatically infer which inducing points are necessary from data. Another problem in the scenario of online machine learning is to define a scheme for removing unnecessary inducing points. This can be done simply by eliminating the oldest ones. More elegantly, the frameworks proposed by McIntire et al. (2016); Bijl et al. (2016) could be employed.

Bibliography

M. Abramowitz. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,*. Dover Publications, Inc., USA, 1974. ISBN 0486612724.

D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety, 2016.

Y. Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, Jan. 2009. ISSN 1935-8237.

J. Bernardo and A. Smith. *Bayesian Theory*, volume 15. 01 2000. ISBN 0 471 49464 X.

H. Bijl, T. B. Schön, J. Wingerden, and M. Verhaegen. Online sparse gaussian process training with input noise. *ArXiv*, abs/1601.08068, 2016.

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

K. Blomqvist, S. Kaski, and M. Heinonen. Deep convolutional gaussian processes, 2018.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Network. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1613–1622. JMLR.org, 2015.

J. Bradshaw, Alexander, and Z. Ghahramani. Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks, July 2017. arXiv:1707.02476.

G. W. Brier. VERIFICATION OF FORECASTS EXPRESSED IN TERMS OF PROBABILITY. *Monthly Weather Review*, 78(1):1–3, 01 1950. ISSN 0027-0644.

F.-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic. Probabilistic integration: A role in statistical computation?, 2015.

D. Burt, C. E. Rasmussen, and M. Van Der Wilk. Rates of convergence for sparse variational Gaussian process regression. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 862–871. PMLR, 2019.

R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian Processes for regression. In *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 3338–3345, 2016.

K. Chalupka, C. K. I. Williams, and I. Murray. A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14, 2013.

Z. Chen, J. Ma, and Y. Zhou. A precise hard-cut em algorithm for mixtures of gaussian processes. pages 68–75, 08 2014. ISBN 978-3-319-09338-3.

Y. Cho and L. K. Saul. Kernel Methods for Deep Learning. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 342–350. Curran Associates, Inc., 2009.

C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20: 273–297, 1995.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, Sept. 2006. ISSN 0018-9448.

L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002. ISSN 0899-7667.

K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. Random feature expansions for deep Gaussian processes. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 884–893. PMLR, 2017.

A. C. Damianou and N. D. Lawrence. Deep Gaussian Processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, volume 31 of *JMLR Proceedings*, pages 207–215. JMLR.org, 2013.

A. Datta, S. Banerjee, A. Finley, and A. Gelfand. On nearest-neighbor gaussian process models for massive spatial data: Nearest-neighbor gaussian process models. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8, 08 2016.

A. P. Dawid. The well-calibrated bayesian. 1982.

M. H. DeGroot and S. E. Fienberg. The comparison and evaluation of forecasters. *The Statistician: Journal of the Institute of Statisticians*, 32:12–22, 1983.

M. P. Deisenroth and J. W. Ng. Distributed gaussian processes. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1481–1490. JMLR.org, 2015.

M. P. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

P. Diggle and P. Ribeiro. *Model-based Geostatistics.* Springer Series in Statistics. Springer, mar 2007. ISBN 0387329072 978-0387329079.

D. K. Duvenaud, O. Rippel, R. P. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 202–210. JMLR.org, 2014.

M. Filippone, M. Zhong, and M. Girolami. A comparative evaluation of stochastic-based inference methods for gaussian process models. *Machine Learning*, 93, 10 2013.

P. A. Flach. Classifier Calibration. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 1–8. Springer US, Boston, MA, 2016.

S. Flaxman, A. G. Wilson, D. B. Neill, H. Nickisch, and A. J. Smola. Fast kronecker inference in gaussian processes with non-gaussian likelihoods. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 607–616. JMLR.org, 2015.

Y. Gal and Z. Ghahramani. Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org, 2016a.

Y. Gal and Z. Ghahramani. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference, Jan. 2016b. arXiv:1506.02158.

Y. Gal and R. Turner. Improving the Gaussian Process Sparse Spectrum Approximation by Representing Uncertainty in Frequency Inputs. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 655–664. JMLR.org, 2015.

Y. Gal, J. Hron, and A. Kendall. Concrete Dropout. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3581–3590. Curran Associates, Inc., 2017.

J. Geweke and M. Keane. Smoothly mixing regressions. *Journal of Econometrics*, 138:252–290, 05 2007.

Z. Ghahramani. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 371:20110553, 02 2013.

J. Ghosh and R. Ramamoorthi. Bayesian nonparametrics. *Springer Series in Statistics*, 16, 01 2011.

E. Gilboa, Y. Saatci, and J. P. Cunningham. Scaling Multidimensional Inference for Structured Gaussian Processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(2):424–436, 2015.

T. Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508, 2002.

T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378, 2007.

G. H. Golub and C. F. Van Loan. *Matrix computations*. The Johns Hopkins University Press, 3rd edition, Oct. 1996. ISBN 080185413.

R. Gramacy. lagp: Large-scale spatial modeling via local approximate gaussian processes in r. *Journal of Statistical Software, Articles*, 72(1):1–46, 2016. ISSN 1548-7660.

R. Gramacy and H. Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103, 11 2007.

R. B. Gramacy and D. W. Apley. Local gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24(2):561–578, 2015.

R. B. Gramacy and B. Haaland. Speeding up neighborhood search in local gaussian process prediction. *Technometrics*, 58(3):294–303, 2016.

R. B. Gramacy and H. K. H. Lee. Adaptive design and analysis of supercomputer experiments. *Technometrics*, 51(2):130–145, 2009.

A. Graves. Practical Variational Inference for Neural Networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.

C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 265–272, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330, International Convention Centre, Sydney, Australia, Aug. 2017. PMLR.

D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2016.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, pages 282–290. AUAI Press, 2013.

J. Hensman, A. Matthews, and Z. Ghahramani. Scalable Variational Gaussian Process Classification. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 351–360. PMLR, 2015a.

J. Hensman, A. G. Matthews, M. Filippone, and Z. Ghahramani. MCMC for Variationally Sparse Gaussian Processes. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1648–1656. Curran Associates, Inc., 2015b.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, Aug. 2002. ISSN 0899-7667.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. ISSN 0899-7667.

N. Hjort, C. Holmes, P. Muller, and S. Walker. *Bayesian Nonparametrics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2010.

Q. M. Hoang, T. N. Hoang, and K. H. Low. A generalized stochastic variational bayesian hyperparameter learning framework for sparse spectrum gaussian process regression, 2016.

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008.

M. Huang, R. Li, H. Wang, and W. Yao. Estimating mixture of gaussian processes by kernel smoothing. *Journal of Business and Economic Statistics*, 32, 05 2014.

R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixture of local expert. *Neural Computation*, 3:78–88, 02 1991.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pages 1339–1344 vol.2, 1993.

A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

H. Kim, B. Mallick, and C. Holmes. Analyzing nonstationary spatial data using piecewise gaussian processes. *Journal of the American Statistical Association*, 100:653 – 668, 2005.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, Apr. 2014.

D. P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the Local Reparameterization Trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.

J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27:75–90, 09 2008.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learning using calibrated regression. In *ICML*, 2018.

M. Kull, T. S. Filho, and P. Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 623–631, Fort Lauderdale, FL, USA, Apr. 2017. PMLR.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast Sparse Gaussian Process Methods: The Informative Vector Machine. In *Advances in Neural Information Processing Systems 15*, pages 625–632. MIT Press, 2002.

M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.

Q. Le, T. Sarlos, and A. Smola. Fastfood - Approximating Kernel Expansions in Loglinear Time. In *30th International Conference on Machine Learning (ICML)*, 2013.

H. Liu, J. Cai, Y. Wang, and Y. Ong. Generalized robust bayesian committee machine for large-scale gaussian process regression. 07 2018a.

H. Liu, Y.-S. Ong, X. Shen, and J. Cai. When gaussian process meets big data: A review of scalable gps, 2018b.

C. Louizos and M. Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1708–1716. JMLR.org, 2016.

C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems 30*, pages 3288–3298. Curran Associates, Inc., 2017.

D. MacKay. Introduction to gaussian processes. 1998.

D. J. MacKay. Bayesian interpolation. *NEURAL COMPUTATION*, 4:415–447, 1991.

D. J. C. Mackay. *Bayesian Methods for Adaptive Models*. PhD thesis, USA, 1992. UMI Order No. GAX92-32200.

D. J. C. Mackay. Bayesian methods for backpropagation networks. In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks III*, chapter 6, pages 211–254. Springer, 1994.

J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2627–2635, Cambridge, MA, USA, 2014. MIT Press.

S. Masoudnia and R. Ebrahimpour. Mixture of experts: A literature survey. *Artificial Intelligence Review*, 42, 08 2014.

M. McIntire, D. Ratner, and S. Ermon. Sparse gaussian processes for bayesian optimization. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, page 517–526, Arlington, Virginia, USA, 2016. AUAI Press. ISBN 9780996643115.

E. Meeds and S. Osindero. An alternative infinite mixture of gaussian process experts. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 883–890. MIT Press, 2006.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Proceedings of the Royal Society of London*, 209:415–446, 1909.

T. P. Minka. Expectation Propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2498–2507. PMLR, 2017.

A. H. Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12(4):595–600, 1973.

I. Murray, R. P. Adams, and D. J. C. MacKay. Elliptical slice sampling. *Journal of Machine Learning Research - Proceedings Track*, 9:541–548, 2010.

A. Naish-Guzman and S. Holden. The generalized FITC approximation. In *Advances in Neural Information Processing Systems 20*, pages 1057–1064. Curran Associates Inc., 2007. ISBN 978-1-60560-352-0.

R. M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer, 1 edition, Aug. 1996. ISBN 0387947248.

T. N. A. Nguyen, A. Bouzerdoum, and S. L. Phung. Variational inference for infinite mixtures of sparse gaussian processes through kl-correction. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2579–2583, 2016.

T. V. Nguyen and E. V. Bonilla. Automated variational inference for gaussian process models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1404–1412. Curran Associates, Inc., 2014a.

T. V. Nguyen and E. V. Bonilla. Fast allocation of gaussian process experts. In *ICML*, 2014b.

A. Niculescu-Mizil and R. Caruana. Predicting Good Probabilities with Supervised Learning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 625–632, New York, NY, USA, 2005. ACM.

A. Panos, P. Dellaportas, and M. K. Titsias. Fully Scalable Gaussian Processes using Subspace Inducing Inputs. 2018. arXiv:1807.02537.

C. Park and D. Apley. Patchwork kriging for large-scale gaussian process regression. *J. Mach. Learn. Res.*, 19(1):269–311, Jan. 2018. ISSN 1532-4435.

C. Park and J. Z. Huang. Efficient computation of gaussian process regression for large spatial data sets by patching local gaussian processes. *J. Mach. Learn. Res.*, 17(1):6071–6099, Jan. 2016. ISSN 1532-4435.

J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3), 1999.

G. Pleiss, J. Gardner, K. Weinberger, and A. G. Wilson. Constant-time predictive distributions for Gaussian processes. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4114–4123. PMLR, 2018.

A. Pourhabib, F. Liang, and Y. Ding. Bayesian site selection for fast Gaussian process regression. *Institute of Industrial Engineers Transactions*, 46(5): 543–555, 2014a.

A. Pourhabib, F. Liang, and Y. Ding. Bayesian site selection for fast gaussian process regression. *IIE Transactions*, 46(5):543–555, 2014b.

M. T. Pratola, H. A. Chipman, J. R. Gattiker, D. M. Higdon, R. McCulloch, and W. N. Rust. Parallel bayesian additive regression trees, 2013.

J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005. ISSN 1532-4435.

J. R. Quilan. *Decision Trees and Multi-Valued Attributes*, page 305–318. Oxford University Press, Inc., USA, 1988. ISBN 0198537182.

A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. *Advances in Neural Information Processing Systems*, 2, 04 2002.

C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

S. Remes, M. Heinonen, and S. Kaski. Non-stationary spectral kernels. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4642–4651. Curran Associates, Inc., 2017.

A. Rudi, L. Carratino, and L. Rosasco. FALKON: An Optimal Large Scale Kernel Method. In *Advances in Neural Information Processing Systems 30*, pages 3888–3898. Curran Associates, Inc., 2017.

W. Rudin. Fourier analysis on groups. 1962.

D. Rullière, N. Durrande, F. Bachoc, and C. Chevalier. Nested kriging predictions for datasets with large number of observations. *Statistics and Computing*, 07 2016.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

Y. Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.

H. Salimbeni and M. P. Deisenroth. Doubly stochastic variational inference for deep gaussian processes. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4591–4602, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

M. Seeger and C. Williams. Fast forward selection to speed up sparse Gaussian process regression. In *Workshop on AI and Statistics 9*. 2003.

M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Artificial Intelligence and Statistics 9*, 2003.

B. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. 1985.

A. J. Smola and P. L. Bartlett. Sparse greedy gaussian process regression. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2005.

E. Snelson and Z. Ghahramani. Variable noise and dimensionality reduction for sparse gaussian processes. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, page 461–468, Arlington, Virginia, USA, 2006. AUAI Press. ISBN 0974903922.

E. Snelson and Z. Ghahramani. Local and global sparse Gaussian process approximations. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007, San Juan, Puerto Rico, March 21-24, 2007*, volume 2 of *JMLR Proceedings*, pages 524–531. JMLR.org, 2007.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.

S. Sun and X. Xu. Variational inference for infinite mixtures of gaussian processes with applications to traffic flow prediction. *Intelligent Transportation Systems, IEEE Transactions on*, 12:466 – 475, 07 2011.

L. S. L. Tan, V. M. H. Ong, D. J. Nott, and A. Jasra. Variational inference for sparse spectrum gaussian process regression. *Statistics and Computing*, 26(6):1243–1261, Sep 2015. ISSN 1573-1375.

M. Titsias and M. Lazaro-Gredilla. Variational inference for mahalanobis distance metrics in gaussian process regression. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 279–287. Curran Associates, Inc., 2013.

M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574. PMLR, 2009.

G.-L. Tran, E. V. Bonilla, J. Cunningham, P. Michiardi, and M. Filippone. Calibrating deep convolutional gaussian processes. volume 89 of *Proceedings of Machine Learning Research*, pages 1554–1563. PMLR, 16–18 Apr 2019.

R. Urtasun and T. Darrell. Sparse probabilistic regression for activity-independent human pose inference. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

M. van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional Gaussian Processes, Sept. 2017.

S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair. Gaussian process modeling of large scale terrain. In *2009 IEEE International Conference on Robotics and Automation*, pages 1047–1053, 2009.

G. Wahba, X. Lin, F. Gao, D. Xiang, R. Klein, and B. Klein. The bias-variance tradeoff and the randomized gacv. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 620–626. MIT Press, 1999.

M. Welling and Y. W. Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. Omnipress, 2011.

C. Williams, C. Rasmussen, A. Schwaighofer, and V. Tresp. Observations on the nyström method for gaussian process prediction. 01 2002.

C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1342–1351, 1998.

A. Wilson and H. Nickisch. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1775–1784. PMLR, 2015.

A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cunningham. Fast kernel learning for multidimensional pattern extrapolation. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3626–3634, Cambridge, MA, USA, 2014. MIT Press.

A. G. Wilson, C. Dann, and H. Nickisch. Thoughts on Massively Scalable Gaussian Processes. 2015. arXiv:1511.01870.

A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing. Stochastic Variational Deep Kernel Learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2586–2594. Curran Associates, Inc., 2016.

J. Yang, P. Yu, and B. Kuo. A nonparametric feature extraction and its application to nearest neighbor classification for hyperspectral image data. *IEEE Transactions on Geoscience and Remote Sensing*, 48(3):1279–1293, 2010.

F. X. Yu, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar. Orthogonal Random Features. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1975–1983. Curran Associates, Inc., 2016.

H. Yu, T. N. Hoang, K. H. Low, and P. Jaillet. Stochastic variational inference for bayesian sparse gaussian process regression, 2017.

S. Yuksel, J. Wilson, and P. Gader. Twenty years of mixture of experts. *Neural Networks and Learning Systems, IEEE Transactions on*, 23:1177–1193, 08 2012a.

S. Yuksel, J. Wilson, and P. Gader. Twenty years of mixture of experts. *Neural Networks and Learning Systems, IEEE Transactions on*, 23:1177–1193, 08 2012b.

B. Zadrozny and C. Elkan. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 694–699, New York, NY, USA, 2002. ACM.

L. Zhao, Z. Chen, and J. Ma. An effective model selection criterion for mixtures of gaussian processes. In X. Hu, Y. Xia, Y. Zhang, and D. Zhao, editors, *Advances in Neural Networks – ISNN 2015*, pages 345–354. Springer International Publishing, 2015a.

L. Zhao, Z. Chen, and J. Ma. An effective model selection criterion for mixtures of gaussian processes. In X. Hu, Y. Xia, Y. Zhang, and D. Zhao, editors, *Advances in Neural Networks – ISNN 2015*, pages 345–354, Cham, 2015b. Springer International Publishing. ISBN 978-3-319-25393-0.

Index

DGP, 4
DTC, 19
FIC, 21
GPs, 3
IPs, 67
KISS-GP, 32
ML, 1
PIC, 22
PITC, 21
RBF, 13
SKI, 33
SOD, 17
SOR, 18
SVGP, 27
SVI, 26
SVM, 16
VSSGP, 38
CNNs, 7
fitc, 20
GPR, 9
GPs, 9
ME, 40
sgpp, 19