



HAL
open science

Advances in Deep Gaussian Processes : calibration and sparsification

Gia-Lac Tran

► **To cite this version:**

Gia-Lac Tran. Advances in Deep Gaussian Processes : calibration and sparsification. Computer Aided Engineering. Sorbonne Université, 2020. English. NNT : 2020SORUS410 . tel-03902494

HAL Id: tel-03902494

<https://theses.hal.science/tel-03902494>

Submitted on 16 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



1

2

**ADVANCES IN DEEP GAUSSIAN PROCESSES:
CALIBRATION AND SPARSIFICATION.**

3

4

Gia-Lac Tran

5

A doctoral dissertation submitted to:

6

SORBONNE UNIVERSITY

7

In Partial Fulfillment of the Requirements for the Degree of:

8

Doctor of Philosophy

9

Specialty : COMPUTER SCIENCE

10

Supervised by: **Prof. Maurizio FILIPPONE**

11

Jury:

Reviewers:

Prof. Marco LORENZI - Inria, Sophia Antipolis - France

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

12

Examiners:

Prof. Pietro MICHIARDI - Eurecom, Biot - France

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

Contents

14	1 Introduction	1
15	1.1 Overview	1
16	1.2 Extensions and Open Problems	4
17	1.3 Outline and Contributions of Thesis	6
18	2 Gaussian Processes for Big Data	8
19	2.1 Overview	8
20	2.2 Gaussian Processes	9
21	2.2.1 Gaussian Processes for Regression	9
22	2.2.2 Covariance function	11
23	2.2.3 Non-Gaussian Likelihoods	14
24	2.2.4 Limitations of Gaussian Processes	14
25	2.3 Inducing Point Approximations	16
26	2.3.1 Prior approximation	16
27	2.3.2 Posterior Approximations	22
28	2.3.3 Structure Exploiting Approximations	27
29	2.4 Random Feature Approximations	33
30	2.4.1 Spectral Representations	34
31	2.4.2 Random featured-based Gaussian Processes.	36
32	2.5 Local Approximation	38
33	3 Calibrating Deep Convolutional Gaussian Processes	41
34	3.1 Introduction	41
35	3.2 Related Work	43
36	3.3 On calibration of Convolutional GPs	45
37	3.4 Proposed Method	46
38	3.4.1 Random Feature Expansions	47
39	3.4.2 End-to-end learning	48
40	3.4.3 Extensions	50
41	3.5 Experiments	52
42	3.5.1 Reliability diagrams	53
43	3.5.2 Extension with Deep GPs	54
44	3.5.3 Knowing when the model does not know	54
45	3.5.4 Extension with the SORF	55
46	3.6 Mathematical details and other experiments	56
47	3.6.1 Random Feature Expansion of the RBF Covariance	56
48	3.6.2 Variational Inference for the Proposed Model	57

49	3.6.3	Variational inference of filters in GPDNN	60
50	3.6.4	Reliability diagrams	60
51	3.7	Conclusions	62
52	4	Local and Global Approximation of Gaussian Processes	66
53	4.1	Introduction	66
54	4.2	Related work and background	68
55	4.2.1	Scalable Variational Gaussian Processes	69
56	4.3	Sparse within Sparse Gaussian Processes	70
57	4.3.1	Lower bound on marginal likelihood	71
58	4.3.2	H-nearest inducing inputs	72
59	4.3.3	Complexity	74
60	4.4	Experiments	75
61	4.4.1	Increasing the number of neighbors	76
62	4.4.2	Increasing the number of inducing points	76
63	4.4.3	Running time	78
64	4.4.4	Large-scale problems with a huge number of IPs	80
65	4.4.5	Comparison to Local GPs	81
66	4.5	Other results	82
67	4.5.1	Various options for H-nearest inducing points selection	82
68	4.5.2	Further visualizations on 1D examples	83
69	4.6	Conclusions	84
70	5	Conclusion	85
71	5.1	Themes and Contributions	85
72	5.2	Future work	86
73	5.2.1	Calibrated GP regression	87
74	5.2.2	Elegant mixtures of CNNs and GPs	87
75	5.2.3	Adaptability to online machine learning	88
76		Bibliography	89

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 mapping 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.

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

124

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

138

139 **Needs of Predictive Uncertainty.** The problem of enhancing the safety
140 of decision-making system by acting on the model’s prediction in an informed
141 manner has obtained a significant attention from the machine learning com-
142 munity (Guo et al., 2017). Predictive uncertainty quantification has a crucial
143 role to strengthen the safety of an AI system (Amodei et al., 2016) by acting
144 on the model’s prediction in an informed manner. This is essential to appli-
145 cations where the consequence of an error is serious, such as in autonomous
146 vehicle control and medical, financial and legal fields. Hence, accurate fit-
147 ting capabilities are no longer the most important aspects for evaluating the
148 model’s effectiveness.

149

150 **Source of Predictive Uncertainty.** Predictive uncertainty is a conflation
151 of several separate factors: model uncertainty, data uncertainty and distribu-
152 tional uncertainty. Model uncertainty or epistemic uncertainty represents the
153 uncertainty in the estimate of model’s parameter given the training data. This
154 uncertainty can be explained away given enough data. Data uncertainty or
155 aleatoric uncertainty comes from the complication in the observations, such as
156 class overlap, label noise, input-dependent noise. As this kind of uncertainty
157 accompanying the nature of data, it is irreducible even if more data are col-
158 lected. Distributional uncertainty appears due to the mismatch between the
159 training and testing distribution.

160
161 **Evaluation of predictive uncertainty.** Evaluating the quality of predictive
162 uncertainties is challenging as the ground-truth uncertainty estimates are un-
163 known. Being motivated by practical applications, there are two aspects that
164 are able to examine the plausibility of predictive uncertainty. The first notion
165 of quality of predictive uncertainty concerns calibration (Dawid, 1982; DeG-
166 root and Fienberg, 1983), which measures the discrepancy between subjective
167 forecast and (empirical) long-run frequencies. Traditionally, the quality of
168 calibration can be numerically assessed by proper scoring rules (Gneiting and
169 Raftery, 2007), such as the Brier score (Brier, 1950). Secondly, the quality
170 of predictive uncertainty is also obtainable using out-of-distribution examples
171 (Hendrycks and Gimpel, 2016). For example, if a model is trained on one
172 dataset, but is tested on a completely different dataset, the predictive un-
173 certainty returned by the model should be high, as testing points would be
174 distant from training points. Recently, the works of approximation of predic-
175 tive uncertainty based upon ensemble learning are robust to calibration as well
176 as the scenarios of data shift (Lakshminarayanan et al., 2017). Alternatively,
177 a plethora of works revolves around the Bayesian formalism (Bernardo and
178 Smith, 2000) with the aim of adapting neural networks to encompass predic-
179 tive uncertainty and give them a probabilistic flavor (Mackay, 1992; Graves,
180 2011; Louizos and Welling, 2016; Blundell et al., 2015).

181
182 **Gaussian Processes.** As alluded earlier, an ideal modeling approach in
183 the era of big data should possess not only a powerful fitting capability but
184 also a firm mechanism to determine predictive uncertainty. Bayesian non-
185 parametric approaches are ideal candidates due to their advantages over flexi-
186 bility and calibrated predictive uncertainty. The philosophies and motivations
187 of this area have been well discussed by a number of authors (Hjort et al.,
188 2010; Ghosh and Ramamoorthi, 2011; Ghahramani, 2013). Gaussian Pro-
189 cesses (GPs) (Rasmussen and Williams, 2006) are an attractive way of doing
190 non-parametric Bayesian modeling. A Gaussian Process is a collection of

191 random variables indexed by a variable in the input domain, such that ev-
192 ery subset of those random variables has a multivariate normal distribution.
193 Thanks to the properties of the multivariate normal distribution, given ob-
194 servations, GPs are able to make inferences as well as predictive uncertainties
195 with a firm mathematical background. In addition to providing uncertainty in
196 predictions, there are also compelling reasons to use GPs, such as the GPs can
197 represent a rich family of functions; also, GPs are protected from overfitting
198 with an appropriate prior on hyperparameters. In practice, GPs achieve state-
199 of-the-art results in a wide spectrum of applications including robotics (Ko
200 and Fox, 2008; Deisenroth and Rasmussen, 2011), geostatistics (Diggle and
201 Ribeiro, 2007), numerics (Briol et al., 2015), active sensing (Guestrin et al.,
202 2005) and optimization (Snoek et al., 2012).

203

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

216 1.2 Extensions and Open Problems

217 In this section, I introduce the extensions and open problems of (Deep) Gaus-
218 sian Processes which will appear in the dissertation.

219

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

229

230 Despite the impressive expressiveness, neural networks access predictive un-
231 certainties via sampling using approaches Welling and Teh (2011); Gal and
232 Ghahramani (2016a); Lakshminarayanan et al. (2017). Unlike neural net-
233 works, GPs directly capture predictive uncertainties with a firm mathematical
234 background. Another advantage of GPs over neural networks is that the prior
235 knowledge about the properties of mapping function, e.g. smoothness, differ-
236 entiability or periodicity, can be added by specifying an appropriate kernel
237 function.

238

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

247

248 **Evaluation of Predictive Uncertainty of Probabilistic Models.** As
249 alluded in the introduction section, predictive uncertainty can be evaluated
250 by inspecting the calibration and out-of-distribution samples. The majority of
251 works accessing predictive uncertainty on NN involve with Bayesian formal-
252 ism Mackay (1992); Graves (2011); Louizos and Welling (2016); Blundell et al.
253 (2015). Along a similar vein, combining CNN and GP is an intuitive way to add
254 probabilistic flavor to CNN Bradshaw et al. (2017); Wilson et al. (2016); van der
255 Wilk et al. (2017). Intuitively, the motivation to impose these Bayesian treat-
256 ments into neural networks is to do a better quantification of uncertainty
257 compared to plain neural networks. Nevertheless, analyzing Bayesian Neural
258 Networks and the combination of neural networks and GPs on predictive un-
259 certainty has not been conducted carefully.

260

261 **Inducing point-based approximation.** GPs Rasmussen and Williams
262 (2006) are well-known because of the predictive uncertainties with a firm
263 mathematical background. Despite being able to underpin a range of al-
264 gorithms for supervised and unsupervised learning, the application of GPs is
265 hindered to the large-scale problems due to the burden of computational and
266 storage cost. Assuming that the input dimensionality D is significantly less
267 than the number of observations N , GPs require the complexities of $\mathcal{O}(N^3)$
268 and $\mathcal{O}(N^2)$ for computation and storage. These costs are sourced from linear
269 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-GP on 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

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

- 315 • **Chapter 3** covers the first primary contribution of this thesis. The
316 study expresses a thorough investigation of the calibration properties
317 of Bayesian Convolutional Neural Networks (CNNs) . Along a similar
318 vein, independently of the works on Bayesian CNNs, there are other
319 attempts to impose a probabilistic formalism to CNNs by integrating
320 CNNs with GPs. Previous work on combining CNNs with GPs has been
321 developed under the assumption that the predictive probabilities of these
322 models are well-calibrated. We show that, in fact, current combinations
323 of CNNs and GPs are miscalibrated. We propose a novel combination
324 that considerably outperforms previous approaches to this aspect, while
325 achieving state-of-the-art performance on image classification tasks.
- 326 • As alluded earlier, inducing point-based idea are a well-known approach
327 to mitigate the computational bottleneck of GPs in the large-scale prob-
328 lems. However, this solution still suffers cubic time complexity to the
329 number of inducing points. Wilson et al Wilson and Nickisch (2015) pro-
330 pose to employ the Kronecker structure on inducing inputs to accelerate
331 the approximation of covariance matrices. The trick also accompanies
332 with significant restrictions on inducing inputs. Besides, the approach
333 only performs well on low-dimensional datasets (Wilson and Nickisch,
334 2015). In **Chapter 4**, we address one limitation of sparse GPs, which is
335 due to the challenge in dealing with a large number of inducing variables
336 without imposing a special structure on the inducing inputs. In partic-
337 ular, we introduce a novel hierarchical prior, which imposes sparsity on
338 the set of inducing variables. The study enables the possibility to use
339 sparse GPs using a large number of inducing points without incurring a
340 prohibitive computational cost.
- 341 • Finally, in **Chapter 5**, we summarize the contributions presented in
342 this thesis. We conclude the thesis by a discussion to an outlook on
343 possible extensions and future work.

Gaussian Processes for Big Data

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

358 2.1 Overview

359 Gaussian Processes (henceforth GPs) which are powerful non-parametric Bayesian
360 models can yield sensible predictions with a small number of available obser-
361 vations. However, it is notorious that GPs suffer from high complexity in terms
362 of both computation and storage with respect to training size N , i.e. $\mathcal{O}(N^3)$
363 and $\mathcal{O}(N^2)$ respectively, so they not the primary choice in datasets with a
364 massive number of data points. To broaden the application of GPs to larger
365 datasets, there is plenty of ideas in the literature that have been proposed
366 and analyzed. According to the groupings mentioned in (Liu et al., 2018b),
367 these approaches are categorized into global and local approximations. While
368 the former approximate the full kernel matrix by a global distillation, the
369 latter abide to the divide-and-conquer concept and make predictions using a
370 local subset of training data. We further split global approximations into sub-
371 categories: Inducing Point-Based Approximation and Random Feature-Based
372 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, $\mathcal{D} = \{(\mathbf{x}_n, y_n) \mid n = 1, \dots, N\}$, 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. $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]^T$, 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:

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

where $\mathbf{m}_{\mathbf{X}}$ are column N -dimensional vector containing mean values at N covariates, i.e. $\mathbf{m}_{\mathbf{X}} = [m(\mathbf{x}_1 \mid \boldsymbol{\zeta}), \dots, m(\mathbf{x}_N \mid \boldsymbol{\zeta})]^T$; 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, $[\mathbf{K}_{\mathbf{X}}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j \mid \boldsymbol{\theta})$.

409

410 **Hyper-parameter optimization.** For convenience sake, we introduce $\boldsymbol{\psi}$
 411 the set of all parameters involving mean function's parameters $\boldsymbol{\zeta}$, kernel pa-
 412 rameters' $\boldsymbol{\theta}$ and variance of likelihood σ_n^2 , i.e. $\boldsymbol{\psi} = (\sigma_n^2, \boldsymbol{\zeta}, \boldsymbol{\theta})$. Given dataset \mathcal{D} ,
 413 Gaussian Processes Regressors are fitted to \mathcal{D} by optimizing hyper-parameter
 414 $\boldsymbol{\psi}$ using the logarithm marginal likelihood function, $\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\psi})$. In gen-
 415 eral, the marginal likelihood can be found by marginalizing over latent random
 416 variables \mathbf{f} .

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

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

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

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

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

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

428

429 **Prediction.** Generally, GPs governs the distribution of a finite-dimensional
 430 vector including latent values at a set of covariates using a multivariate nor-
 431 mal distribution. Therefore, the joint distribution of training latent values, \mathbf{f} ,
 432 and the testing latent values, \mathbf{f}_* , according to the GP prior is:

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

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

438

439 Remind that, in regression, the likelihood of observable targets given training
 440 latent values are intuitively assumed to be a Gaussian with the variance of σ_n^2 ,
 441 $p(y_n | f_n) = \mathcal{N}(y_n | f_n, \sigma_n^2)$. It means that the functions for observable targets
 442 can be modeled as a noisy version of latent function f a Gaussian noise with
 443 variance of σ_n^2 , $y(\mathbf{x}_n) = f(\mathbf{x}_n) + \varepsilon$, where ε follows $\mathcal{N}(\varepsilon | 0, \sigma_n^2)$. Assuming
 444 additive independent identically distributed Gaussian noise with variance σ_n^2 ,
 445 the prior on the noisy observations becomes:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{y} | \mathbf{m}_{\mathbf{X}}, \mathbf{K}_{\mathbf{X}} + \sigma_n^2 \mathbf{I}). \quad (2.6)$$

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

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

448 To get the posterior distribution over function, we need to restrict this joint
 449 prior distribution to contain only those functions which agree with the ob-
 450 served data points. By virtue of the nice properties of the multivariate normal
 451 distribution, the operation of eliminating those violating the available obser-
 452 vations is extremely simple, corresponding to conditioning the joint Gaussian
 453 prior distribution on the observations to give:

$$\mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\psi} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{f}_*}, \boldsymbol{\Sigma}_{\mathbf{f}_*}), \text{ where} \quad (2.8)$$

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

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

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

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

456 2.2.2 Covariance function

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

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

467

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

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

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

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

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

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

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

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

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

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

- 501 • Squared Exponential Covariance Function.

502

503 The square exponential function or Radial Basis Function (RBF) ker-
 504 nel has the form:

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

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

- 507 • The Matérn Covariance Function.

508

509 The Matérn class of covariance functions is given by

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

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

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

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

513 **Automatic Relevance Determination Kernel.** The kernel functions men-
 514 tioned above are called isotropic where the flexibility of kernel function is
 515 indicated by a lengthscale parameter, l . To enhance the flexibility of ker-
 516 nel function, we augment D length-scale parameters, l_1, \dots, l_D accompanying
 517 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:

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

, where $\Lambda = \text{Diag}[l_1^2, \dots, l_D^2]^T$. 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 $p(\mathbf{y} | \mathbf{f})$ is conjugate to the Gaussian prior $p(\mathbf{f})$. 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 $p(\mathbf{y} | \mathbf{X}, \boldsymbol{\psi}, \mathbf{y})$. Take GPs regression with zero mean prior as an example, the gradients of marginal likelihood with respect to parameter ψ_i is computed as:

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

550 The computation of gradients involves with solving the linear system, i.e.
 551 $\mathbf{K}_{\sigma_n^2}^{-1}\mathbf{y}$ where $\mathbf{K}_{\sigma_n^2}$ is $N \times N$ covariance matrix with additive noise and \mathbf{y} is N -
 552 dimensional column vector of outputs, where N is the number of data points.
 553 Practically, this linear system is solved by using Cholesky decomposition to
 554 factorize the symmetric positive definite kernel matrix $\mathbf{K}_{\sigma_n^2}$ into $\mathbf{L}\mathbf{L}^T$, where
 555 \mathbf{L} is a lower triangular matrix. Generally, factorization with Cholesky decom-
 556 position necessitates $\mathcal{O}(N^3)$ operations. The calculation of the trace terms
 557 appearing in gradient formula also need $\mathcal{O}(N^3)$ operations. In the progress
 558 of computing the gradients, the lower triangular matrix \mathbf{L} must be cached.
 559 Therefore, the storage cost of the training phase is $\mathcal{O}(N^2)$.

560

561 Besides the cubic complexity in the training phase, the computational cost of
 562 GPs inference also depends on the training size. On inspection of the predictive
 563 distribution given from equation 2.8 to 2.10, we can observe that evaluating
 564 this expression also involves the inversion of $N \times N$ kernel matrix. Theoretic-
 565 ically, the computational cost for GP inference is also $\mathcal{O}(N^3)$. However, in
 566 practice, the inversion of $\mathbf{K}_{\sigma_n^2}$ and the vector which is the multiplication of
 567 $\mathbf{K}_{\sigma_n^2}^{-1}$ and $\mathbf{y} - \mathbf{m}_{\mathbf{X}}$ can be recorded after the training phase. Therefore, the
 568 computational costs of predictive mean and predictive variance at an unseen
 569 data point are $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$. As discussed, the likelihood mapping the
 570 latent values to observation is not obligated to be a Gaussian, as the case of
 571 classification. Under these conditions, the computation of the marginal likeli-
 572 hood as well as inference is no longer analytic, and further approximations are
 573 required, and computational budgets required in these case is identical to GP
 574 regression. Due to the dependence of computational complexity on training
 575 size, GPs are hindered to large-scale problems.

576

577 To strengthen the scalability, while retaining the desired prediction quality, a
 578 large number of scalable GPs have been proposed. According to (Liu et al.,
 579 2018b), these scalable approaches are sorted into two main categories: local
 580 and global approximation. Local approximations arising from the divide-and-
 581 conquer concept focus on the local area of input spaces (Gramacy and Lee,
 582 2007; Yuksel et al., 2012b; Masoudnia and Ebrahimpour, 2014; Rasmussen
 583 and Ghahramani, 2002; Sun and Xu, 2011; Hinton, 2002; Deisenroth and Ng,
 584 2015; Rulli ere et al., 2016; Liu et al., 2018a). Whereas global approximations
 585 replace kernel matrix $\mathbf{K}_{\mathbf{X}}$ by a compact representation reducing the burden
 586 of computation. The substitution is done through global distillation which
 587 can be accomplished by several ways, e.g. use a small subset of training data
 588 (Chalupka et al., 2013), or remove uncorrelated entries in $\mathbf{K}_{\mathbf{X}}$ using sparse
 589 kernel (Gneiting, 2002), or employ low-rank representation (Nystr om approx-
 590 imation) (Hensman et al., 2013; Qui onero Candela and Rasmussen, 2005;

591 Titsias, 2009; Wilson and Nickisch, 2015).

592

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

609 2.3 Inducing Point Approximations

610 2.3.1 Prior approximation

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

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

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

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

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

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

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

628 Here, $\hat{\mathbf{X}}$ generally indicates training inputs \mathbf{X} and testing inputs \mathbf{X}_* . As-
629 suming that \mathbf{A} and \mathbf{B} are the matrices constructed by concatenating co-
630 variates likewise \mathbf{X} and \mathbf{X}_* , we define $\mathbf{K}_{\mathbf{A}, \mathbf{B}}$ as a cross covariance matrix
631 whose element in the i, j position is the covariance between the i -th co-
632 variate in \mathbf{A} and j -th covariate in \mathbf{B} . We also introduce the shorthand no-
633 tation $\mathbf{Q}_{\mathbf{A}, \mathbf{B}} = \mathbf{K}_{\mathbf{A}, \mathbf{Z}} \mathbf{K}_{\mathbf{Z}, \mathbf{Z}}^{-1} \mathbf{K}_{\mathbf{Z}, \mathbf{B}}$ which can be seen as an approximation to
634 $\mathbf{K}_{\mathbf{A}, \mathbf{B}}$ using inducing inputs \mathbf{Z} . For simplicity, we use the Gaussian likelihood
635 $p(\mathbf{y} | \mathbf{f}) = \mathcal{N}(\mathbf{y} | \mathbf{f}, \sigma_n^2 \mathbf{I})$. The predictive latent distributions $p(\mathbf{f}_* | \mathbf{y})$ can be
636 written in a closed-form using Gaussian density:

$$p(\mathbf{f}_* | \mathbf{y}) = \mathcal{N}(\mathbf{f}_* | \mathbf{K}_{\mathbf{X}_*, \mathbf{X}} (\mathbf{K}_{\mathbf{X}, \mathbf{X}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{K}_{\mathbf{X}_*, \mathbf{X}_*} - \mathbf{K}_{\mathbf{X}_*, \mathbf{X}} (\mathbf{K}_{\mathbf{X}, \mathbf{X}})^{-1} \mathbf{K}_{\mathbf{X}, \mathbf{X}_*} + \sigma_n^2 \mathbf{I}). \quad (2.27)$$

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

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

640 Following the unifying view mentioned by Quiñonero Candela and Rasmussen
641 (2005), a particular algorithm complying with the idea of prior approximation
642 corresponds to different additional assumptions about the two approximate
643 inducing conditionals $q(\mathbf{f} | \mathbf{u})$ and $q(\mathbf{f}_* | \mathbf{u})$ appearing in the approximation
644 defined in 2.28. The method PIC (Snelson and Ghahramani, 2007) mentioned
645 at the end of the section is also an extension of the idea by using another way
646 to approximate the joint prior $p(\mathbf{f}_*, \mathbf{f})$.

647

648 **Subset of Data.** The most straightforward approach to reduce the com-
649 putational burden of GPs, which stems from the inverse of the kernel matrix
650 $\mathbf{K}_{\mathbf{X}}$, is to work on subsets of the data (henceforth SOD), \mathcal{D}_{SOD} for the whole
651 training points, \mathcal{D} , i.e. simply speaking, we use $\mathbf{K}_{\mathbf{X}_{\text{SOD}}}$ instead of $\mathbf{K}_{\mathbf{X}}$. By
652 restricting the number of data point M in \mathbf{X}_{SOD} to be less than the total num-
653 ber of observations, N , the computational cost will decrease from $\mathcal{O}(N^3)$ to

654 $\mathcal{O}(M^3)$. In case \mathbf{X}_{SOD} is specified in an appropriate manner, the approaches
 655 of SOD will produce reasonable predictive distributions. Otherwise and most
 656 often, SOD yields overconfident predictions. On the inspection of the selection
 657 of \mathcal{D}_{SOD} , one could, for example, randomly choose M data points, use clus-
 658 tering techniques to divide the training data to M subsets and then choose
 659 the centroids as representative for all the whole data sets, or employ online
 660 learning scheme with criteria based on information theoretic principles, i.e.
 661 differential entropy (Lawrence et al., 2002), to choose active data points se-
 662 quentially.

663

664 Turning to the unifying view mentioned above, \mathbf{u} and \mathbf{f} are replaced by \mathbf{f}_{sod}
 665 which are the latent values of subset input \mathcal{D}_{SOD} . SOD also uses the true
 666 testing conditional distribution instead of its approximation, i.e. $q(\mathbf{f}_* | \mathbf{u}) =$
 667 $p(\mathbf{f}_* | \mathbf{f}_{\text{sod}})$. The joint prior turns out to be:

$$p(\mathbf{f}_*, \mathbf{f}) \rightarrow p(\mathbf{f}_*, \mathbf{f}_{\text{sod}}) = \int p(\mathbf{f}_* | \mathbf{f}_{\text{sod}}) p(\mathbf{f}_{\text{sod}}) \quad (2.29)$$

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

$$q_{\text{SOR}}(\hat{\mathbf{f}} | \mathbf{u}) = \mathcal{N}(\mathbf{f} | \mathbf{K}_{\hat{\mathbf{x}}, \mathbf{z}} \mathbf{K}_{\mathbf{z}, \mathbf{z}}^{-1} \mathbf{u}, \mathbf{0}). \quad (2.30)$$

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

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

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

$$q_{\text{SOR}}(\mathbf{f}_* | \mathbf{y}) = \mathcal{N}(\mathbf{f}_* | \mathbf{Q}_{\mathbf{x}_*, \mathbf{x}} (\mathbf{Q}_{\mathbf{x}, \mathbf{x}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \quad (2.32)$$

$$\mathbf{Q}_{\mathbf{x}_*, \mathbf{x}_*} - \mathbf{Q}_{\mathbf{x}_*, \mathbf{x}} (\mathbf{Q}_{\mathbf{x}, \mathbf{x}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{Q}_{\mathbf{x}, \mathbf{x}_*})$$

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

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

684

685 **Deterministic Training Conditional.** According to the analysis of Williams
 686 et al. (2002), SOR can yield negative predictive variances due to the approxi-
 687 mation of the full covariance matrix using the Nyström method. In order to
 688 avoid these nonsensical predictive variances, Seeger and Williams (2003) pro-
 689 posed a novel sparse approximation to Gaussian Processes Regression. The
 690 approach mainly relies on a likelihood approximation, based on the projection
 691 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 pro-
 692 jection, this approach is called to Deterministic Training Conditional (DTC).

$$p(\mathbf{y} | \mathbf{f}) = \mathcal{N}(\mathbf{y} | \mathbf{f}, \sigma_n^2 \mathbf{I}) \approx q_{\text{DTC}}(\mathbf{y} | \mathbf{u}) = \mathcal{N}(\mathbf{y} | \mathbf{K}_{\mathbf{x}, \mathbf{z}} \mathbf{K}_{\mathbf{z}, \mathbf{z}}^{-1} \mathbf{u}, \sigma_n^2 \mathbf{I}) \quad (2.33)$$

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

$$q_{\text{DTC}}(\mathbf{f} | \mathbf{u}) = q_{\text{SOR}}(\mathbf{f} | \mathbf{u}) = \mathcal{N}(\mathbf{f} | \mathbf{K}_{\mathbf{x}, \mathbf{z}} \mathbf{K}_{\mathbf{z}, \mathbf{z}}^{-1} \mathbf{u}, \mathbf{0}). \quad (2.34)$$

$$q_{\text{DTC}}(\mathbf{f}_* | \mathbf{u}) = p(\mathbf{f}_* | \mathbf{u}) = \mathcal{N}(\mathbf{f}_* | \mathbf{K}_{\mathbf{x}_*, \mathbf{z}} \mathbf{K}_{\mathbf{z}, \mathbf{z}}^{-1} \mathbf{u}, \mathbf{K}_{\mathbf{x}_*, \mathbf{x}_*} - \mathbf{Q}_{\mathbf{x}_*, \mathbf{x}_*}). \quad (2.35)$$

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

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

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

$$q_{\text{DTC}}(\mathbf{f}_* | \mathbf{y}) = \mathcal{N}(\mathbf{f}_* | \mathbf{Q}_{\mathbf{x}_*, \mathbf{x}} (\mathbf{Q}_{\mathbf{x}, \mathbf{x}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{K}_{\mathbf{x}_*, \mathbf{x}_*} - \mathbf{Q}_{\mathbf{x}_*, \mathbf{x}} (\mathbf{Q}_{\mathbf{x}, \mathbf{x}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{Q}_{\mathbf{x}, \mathbf{x}_*}) \quad (2.37)$$

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

710 of active locations and kernel hyper-parameters.

711

712 Integrating SGPP into the unifying framework, we can observe clearly the
 713 differences in the formalism between SGPP and SOR and DTC. While the like-
 714 lihood variance of DTC is characterized by only the noise variance, the likeli-
 715 hood variance of SGPP also takes into account the residual difference between
 716 $\text{Diag}(K_{\mathbf{X},\mathbf{X}})$ and $\text{Diag}(\mathbf{Q}_{\mathbf{X},\mathbf{X}})$. SGPP assumes that the auxiliary variables \mathbf{u} in-
 717 duces the relation of training latent variables \mathbf{f} . Due to this assumption, SGPP
 718 can be called Fully Independent Training Conditional (FITC) approximation.
 719 The approximation to the likelihood as well as the variational distribution of
 720 training and testing latent values given \mathbf{u} also relies on the projection as in
 721 DTC, but the predictive variances are more sophisticated than DTC.

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

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

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

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

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

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

$$q_{\text{FITC}}(\mathbf{f}_* | \mathbf{y}) = \mathcal{N}(\mathbf{f}_* | \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}, \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}}(\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \mathbf{\Lambda} + \sigma_n^2\mathbf{I})^{-1}\mathbf{Q}_{\mathbf{X},\mathbf{X}_*}). \quad (2.42)$$

731 Observe the approximation to joint prior in FITC defined in the equation
 732 2.41, we realize that the training and testing covariance are constructed het-
 733 erogeneously. Therefore, the approximation FITC does not comply with the
 734 strict definition of GPs where the covariance for all points must be com-
 735 puted by identical manners. In contrast, if the assumption of conditional

736 independence given active points is extended to the testing case, FITC turns
 737 into another approach which is logically called Fully Independent Conditional
 738 (FIC) . FIC is equivalent to Gaussian Processes with the covariance function
 739 $k_{\text{FIC}}(\mathbf{x}_i, \mathbf{x}_j) = k_{\text{SOR}}(\mathbf{x}_i, \mathbf{x}_j) + \delta_{i,j} (k(\mathbf{x}_i, \mathbf{x}_j) - k_{\text{SOR}}(\mathbf{x}_i, \mathbf{x}_j))$, where $\delta_{i,j}$ is Kro-
 740 necker delta function. The prior and predictive distribution implied by FIC
 741 is:

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

742

$$q_{\text{FIC}}(\mathbf{f}_* | \mathbf{y}) = \mathcal{N}(\mathbf{f}_* | \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} (\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \Lambda + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}_*} + \Lambda_* - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} (\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \Lambda + \sigma_n^2 \mathbf{I})^{-1} \mathbf{Q}_{\mathbf{X},\mathbf{X}_*}). \quad (2.44)$$

743 **Partially Independent (Training) Conditional.** Having compared the
 744 predictive distribution of DTC and FITC defined in equation 2.37 and 2.42, it
 745 is obvious that FITC is an improvement of DTC by remaining the exact diag-
 746 onal elements of the covariance matrix. Relying on the unifying framework,
 747 Quiñonero Candela and Rasmussen (2005) have proposed a further improved
 748 approximation compared to FITC by extending the training conditional to have
 749 a block of diagonal covariance and remaining the exact testing covariance as
 750 defined in equation 2.26. Due to the usage of diagonal block covariance on
 751 training conditional, the approximation is called Partially Independent Train-
 752 ing Conditionals (PITC) .

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

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

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

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

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

$$q_{\text{PITC}}(\mathbf{f}_* | \mathbf{y}) = \mathcal{N}(\mathbf{f}_* | \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} (\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \tilde{\Lambda} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{Q}_{\mathbf{X}_*,\mathbf{X}} (\mathbf{Q}_{\mathbf{X},\mathbf{X}} + \tilde{\Lambda} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{Q}_{\mathbf{X},\mathbf{X}_*}). \quad (2.48)$$

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

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

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

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

779 , where

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

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

782 2.3.2 Posterior Approximations

783 **Weakness of Prior Approximations.** As alluded to previously, the
 784 aforementioned algorithms complying with the idea of prior approximation
 785 operate as an exact Gaussian Processes with a particular kernel function
 786 or an approximation to covariance matrices, i.e. SOR (Smola and Bartlett,
 787 2001) and FIC (Snelson and Ghahramani, 2006) and PIC (Snelson and Ghahra-
 788 mani, 2007). Suppose we would like to employ M inducing variables which
 789 are latent values at some auxiliary inputs \mathbf{Z} to approximate the GP prior.

790 The quality of these sparse approximations depends on the optimization of \mathbf{Z}
 791 and hyper-parameters, i.e. kernel’s parameters and variance noise (for Gaus-
 792 sian likelihood). An approximation to the true marginal likelihood defined in
 793 2.1 allows us to select \mathbf{Z} and other hyper-parameter using a gradient-based
 794 iterative method. For example, consider a zero-mean GP, Projected Pro-
 795 cess approximation (PP) (Seeger et al., 2003) and Sparse Gaussian Processes
 796 using Pseudo-points (SGPP) (Snelson and Ghahramani, 2006) following the
 797 idea of prior approximation replace the logarithm of GP marginal likelihood
 798 $F_{\text{GP}} = \log [p(\mathbf{y} | \mathbf{X})]$ by F_{PP} and F_{SGPP} :

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

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

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

799 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 approxi-
 800 mation to $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ using inducing inputs \mathbf{Z} . $\mathbf{\Lambda} = \text{Diag}[\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{Q}_{\mathbf{X},\mathbf{X}}]$ is the difference
 801 on diagonal elements between the true kernel values and approximated ones.
 802 Observe F_{PP} and F_{SGPP} , the covariance of approximate marginal likelihood are
 803 parameterized by inducing inputs \mathbf{Z} . While it is tempting to think that the
 804 introduction of \mathbf{Z} in kernel function improves the representational power of
 805 approximate GPs, the highly-parameterized form probably lead to an over-
 806 fitting problem because the continuous optimization of F_{PP} and F_{SGPP} with
 807 respect to \mathbf{Z} does not reliably approximate true GP.

808
 809 **Main idea of Posterior Approximations.** In order to deal with the
 810 lack of consideration of the convergence between true GP and approximate
 811 ones, we intuitively find \mathbf{Z} by minimizing the *distance* of the approximated
 812 predictive distributions produced by the inducing points and the true ones.
 813 Further, the idea also allows us to access the divergence between the true GP
 814 and sparse approximation to GP. Since both of predictive distribution $p(\mathbf{f}_* | \mathbf{y})$
 815 and posterior $p(\mathbf{f} | \mathbf{y})$ are conditional prior given observations, the selection
 816 of \mathbf{Z} based upon the idea is equivalent to minimize the Kullback-Leibler diver-
 817 gence between the approximate posterior $q(\mathbf{f})$ and the true posterior $p(\mathbf{f} | \mathbf{y})$.

818
 819 Starting from the true GP conditional prior over arbitrary auxiliary variable
 820 \mathbf{v} given observations \mathbf{y} , we construct the approximate GP posterior using M
 821 inducing points. We can express the conditional prior $p(\mathbf{v} | \mathbf{y})$ by integrating
 822 out inducing variables \mathbf{u} and training latent values \mathbf{f} as follows:

$$p(\mathbf{v} | \mathbf{y}) = \int p(\mathbf{v} | \mathbf{u}, \mathbf{f}) p(\mathbf{f} | \mathbf{u}, \mathbf{y}) p(\mathbf{u} | \mathbf{y}) d\mathbf{f} d\mathbf{u}. \quad (2.55)$$

823 By assuming \mathbf{u} completely capture the relation between \mathbf{v} and \mathbf{f} , it holds that
 824 $p(\mathbf{v} | \mathbf{u}, \mathbf{y}) = p(\mathbf{v} | \mathbf{u})$. Thanks to the assumption of conditional indepen-
 825 dence, the variable \mathbf{f} only appears in $p(\mathbf{f} | \mathbf{u}, \mathbf{y})$, and therefore, \mathbf{f} is canceled
 826 out as $\int p(\mathbf{f} | \mathbf{u}, \mathbf{y}) d\mathbf{f} = 1$. Subsequently, the above $p(\mathbf{v} | \mathbf{y})$ can be written
 827 as $q(\mathbf{v})$:

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

828 where $q(\mathbf{v}) = p(\mathbf{v} | \mathbf{y})$ and $q(\mathbf{u}) = p(\mathbf{u} | \mathbf{y})$. Practically speaking, it is in-
 829 feasible to find inducing variables \mathbf{u} which are sufficient statistics for the pa-
 830 rameters \mathbf{f} . Thus, $q(\mathbf{v})$ should be expected as an approximation to $p(\mathbf{v} | \mathbf{y})$.
 831 Subsequently, the $q(\mathbf{u})$ can be represented by a parameterized form.

832
 833 Since the joint variable $[\mathbf{z}, \mathbf{y}]^T$ and $[\mathbf{z}, \mathbf{u}]^T$ follow a GP, the conditional pri-
 834 ors of $p(\mathbf{z} | \mathbf{y})$ and $p(\mathbf{z} | \mathbf{u})$ are also Gaussian densities. Intuitively, $q(\mathbf{z})$
 835 which is the approximation to $p(\mathbf{z} | \mathbf{y})$ should be also a Gaussian. Thanks to
 836 the equation 2.56 and the conjugacy properties, we see that assuming $q(\mathbf{u})$ a
 837 variational Gaussian distribution defined by a mean vector \mathbf{m} and covariance
 838 matrix \mathbf{S} turns $q(\mathbf{v})$ to be a Gaussian. Introducing $\tilde{\mathbf{X}}$ as the indices of \mathbf{v} , we
 839 can express $q(\mathbf{v})$ under a closed form:

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

840 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
 841 function values, $q(\mathbf{v})$ can be perceived as an approximation to GP posterior
 842 $q(\mathbf{f})$ or predictive distributions $q(\mathbf{f}_*)$.

843
 844 Turning to the idea of posterior approximation, all parameters $\boldsymbol{\theta}$, e.g. in-
 845 ducing inputs or hyper-parameters, are selected to minimize the Kullback-
 846 Leibler divergence between the approximate posterior $q(\mathbf{f})$ and the true pos-
 847 terior $p(\mathbf{f} | \mathbf{y})$. Equivalently, we can minimize a distance between the aug-
 848 mented variational posterior $q(\mathbf{f}, \mathbf{u})$ defined in equation 2.56, i.e. $q(\mathbf{f}, \mathbf{u}) =$
 849 $p(\mathbf{f} | \mathbf{u})\phi(\mathbf{u})$ and the augmented true posterior $p(\mathbf{f}, \mathbf{u} | \mathbf{y})$.

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \text{KL}[q(\mathbf{f}, \mathbf{u}) || p(\mathbf{f}, \mathbf{u} | \mathbf{y})] \quad (2.58)$$

850 Taking further analysis, we see that $\text{KL}[q(\mathbf{f}, \mathbf{u}) || p(\mathbf{f}, \mathbf{u} | \mathbf{y})]$ can be repre-
 851 sented as:

$$\text{KL}[q(\mathbf{f}, \mathbf{u}) || p(\mathbf{f}, \mathbf{u} | \mathbf{y})] = \log[p(\mathbf{y})] - \mathbb{E}_{q(\mathbf{f}, \mathbf{u})} \log \left[\frac{p(\mathbf{f}, \mathbf{u}, \mathbf{y})}{q(\mathbf{f}, \mathbf{u})} \right]. \quad (2.59)$$

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

$$F_q = E_{q(\mathbf{f}, \mathbf{u})} \log \left[\frac{p(\mathbf{f}, \mathbf{u}, \mathbf{y})}{q(\mathbf{f}, \mathbf{u})} \right] = \int p(\mathbf{f} | \mathbf{u}) q(\mathbf{u}) \log \left[\frac{p(\mathbf{y} | \mathbf{f}) p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{f} d\mathbf{u}. \quad (2.60)$$

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

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

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

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

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

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

866 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
867 to $\mathbf{K}_{\mathbf{X}, \mathbf{X}}$ using inducing inputs \mathbf{Z} , and $\mathbf{\Lambda} = \text{Diag}[\mathbf{K}_{\mathbf{X}, \mathbf{X}} - \mathbf{Q}_{\mathbf{X}, \mathbf{X}}]$ is the differ-
868 ence on diagonal elements between the true kernel values and approximated
869 ones. Observe the approximation to GP marginal likelihood of the approach of
870 Projected Process Approximation (PP) or Deterministic Training Conditions
871 (DTC) defined in 2.53, we can rewrite F_{SGPR} in terms of F_{PP} :

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

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

878 also prevents SGPR from overfitting.

879

880 **Stochastic Variational Inference for Gaussian Processes.** A downside
881 of SGPR proposed by Titsias (2009) is that the computational and storage
882 cost depends on the training size N linearly. On the inspection of the bound
883 F_{SGPR} defined in equation 2.64, each training iteration of SGPR requires the
884 budget of $\mathcal{O}(NM^2)$ for computation and $\mathcal{O}(NM)$. These costs come from the
885 linear algebraic operation appearing the computation of $\mathbf{Q}_{\mathbf{x},\mathbf{x}}$, i.e. the matrix
886 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
887 reduction of SGPR on computation and memory requirement are impressive,
888 these demands are quickly prohibitive for big data, where the training size N
889 reaches to many millions or billions.

890

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

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

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

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

904 Apart from accelerating the computation cost by applying stochastic varia-
905 tional 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:

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

, where $q(f_i)$ is calculated as $q(\mathbf{v})$ defined in equation 2.57. In case the likelihood $p(y_i | f_i)$ 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 $p(y_i | f_i) = \mathcal{N}(y_i | f_i, \sigma_n^2)$, the gradients of logarithm of the marginal likelihood $p(\mathbf{y} | \mathbf{X})$ with respect to an arbitrary trainable parameter ψ is as follows:

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

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 $\mathcal{O}(N^3)$ (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

932 $\mathbf{K}_{\mathbf{X},\mathbf{X}}$. A well-known approach following the idea of structural exploitation is
 933 to use Kronecker product with the assumption of grid-structure dataset and
 934 tensor product kernel (Saatçi, 2011; Gilboa et al., 2015).

935
 936 **Grid-structured data.** According to the exposition of Chapter 5 in Saatçi’s
 937 dissertation, we assume all input points \mathbf{X} are located on a Cartesian grid,
 938 i.e.

$$\mathbf{X} = \mathbf{X}_1 \times \cdots \times \mathbf{X}_D \quad (2.70)$$

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

956
 957 **Tensor product kernel.** In this section, the covariance functions are as-
 958 sumed to be tensor product kernels, which compute the covariance as a sepa-
 959 rable product over dimensions. Introducing two D -dimensional covariates \mathbf{x}_i
 960 and \mathbf{x}_j belonging to the grid-structure input space \mathbf{X} mentioned above, the
 961 covariance between \mathbf{x}_i and \mathbf{x}_j can be written as:

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

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

964
 965 **Algebraic advantages of the Kronecker method.** Introducing \mathbf{A} as
 966 $m \times n$ matrix and \mathbf{B} as $p \times q$ matrix, the Kronecker product of \mathbf{A} and \mathbf{B} ,

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

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

968 , more explicitly:

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

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

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

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

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

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

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

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

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

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

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

974

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

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

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

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

982 It is infeasible to access the logarithm marginal likelihood of GP regression
 983 on \mathbf{X} containing $N = \prod_{d=1}^D G_d$ points due to the computational bottlenecks
 984 from the algebraic operations, i.e. the inversion and matrix-vector multipli-
 985 cation $(\mathbf{K}_{\mathbf{X}, \mathbf{X}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$ and logarithm of determinant $\log |\mathbf{K}_{\mathbf{X}, \mathbf{X}} + \sigma_n^2 \mathbf{I}|$. The
 986 original computational and storage cost are $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ respectively.
 987 Due to the nice properties of Kronecker product, the complexity of learning
 988 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
 989 section, I will analyze and explain why Kronecker product can lead to the
 990 improvements.

991
 992 These reductions come from the fast computation of eigendecomposition $\mathbf{K}_{\mathbf{X}, \mathbf{X}} =$
 993 $\mathbf{Q}\mathbf{V}\mathbf{Q}^T$, where \mathbf{V} is the diagonal matrix constructed by corresponding eigen-
 994 values v_i , and \mathbf{Q} is the square matrix whose i -th column is the eigenvector
 995 q_i of $\mathbf{K}_{\mathbf{X}, \mathbf{X}}$, and therefore, \mathbf{Q} is guaranteed to be an orthogonal matrix, and
 996 consequently, $\mathbf{Q}^{-1} = \mathbf{Q}^T$. Since $\mathbf{K}_{\mathbf{X}, \mathbf{X}}$ can be expressed by Kronecker product,
 997 the computation for matrices \mathbf{Q} and \mathbf{V} is accelerated by separately carrying
 998 out the eigendecomposition of $\mathbf{K}_1(\mathbf{X}_1, \mathbf{X}_1), \dots, \mathbf{K}_D(\mathbf{X}_D, \mathbf{X}_D)$. Denoting \mathbf{Q}_d
 999 as matrix containing eigenvectors of $\mathbf{K}_d(\mathbf{X}_d, \mathbf{X}_d)$ and \mathbf{V}_d as a diagonal ma-
 1000 trix 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}
 1001 and \mathbf{V} can be expressed as Kronecker products by using the Mixed-product
 1002 property defined at 2.76. Actually, \mathbf{V}_d and \mathbf{V} are diagonal matrices, and \mathbf{V}
 1003 are constructed by concatenating diagonal elements of \mathbf{V}_d .

$$\mathbf{Q} = \mathbf{Q}_1 \otimes \dots \otimes \mathbf{Q}_D, \text{ and } \mathbf{V} = \text{Diag}\left(\text{Diag}(\mathbf{V}_1)^T, \dots, \text{Diag}(\mathbf{V}_D)^T\right) \quad (2.84)$$

1004 Due to the *Vec* property mentioned at 2.81, the fast matrix vector multipli-
 1005 cation are enabled by using the Algorithm `kron-mvn` mentioned in Saatçi's
 1006 dissertation (Saatçi, 2011).

1007
 1008 In order to analyze the complexity of `kron_mvm` conveniently, I assume all
 1009 \mathbf{A}_d have the same dimensions. Similarly, the algorithm `kron_mvm` also works
 1010 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
 1011 iterative steps appear in the loop, the computational cost mainly relies on the
 1012 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 $\mathbf{A}_1, \dots, \mathbf{A}_D$, N -dimensional vector \mathbf{b} where $N = G^D$

Output: $\boldsymbol{\alpha} = (\otimes_{d=1}^D \mathbf{A}_d) \mathbf{b}$.

```

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

```

1013 is repeated D times, therefore, the ultimate budget is $\mathcal{O}\left(DN^{1+\frac{1}{D}}\right)$.

1014

1015 Turning to the matrix-vector multiplication $(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$ appearing in
 1016 the logarithm of marginal likelihood, it can be rewritten in terms of \mathbf{Q} and
 1017 \mathbf{V} . Thanks to the property of *Transpose* defined at 2.78, $(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$
 1018 can be further represented by Kronecker product:

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

1019 With the above expression, the matrix vector multiplication can be solved
 1020 efficiently using the following steps:

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

1021 where kron_mvm is a procedure detailed in Algorithm 1. Remind that \mathbf{V} is di-
 1022 agonal matrix containing the eigenvalues of block covariances $\{\mathbf{K}_d(\mathbf{X}_d, \mathbf{X}_d)\}_{d=1}^D$,
 1023 the matrix $\mathbf{V} + \sigma_n^2 \mathbf{I}$ is also diagonal, and, therefore, its inversion can be com-
 1024 puted with linear complexity.

1025

1026 The fast eigendecomposition of $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ also speeds up the computation of log-
 1027 arithm of determinant of $\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}$. Denoting v_1, \dots, v_N the diagonal ele-
 1028 ments of \mathbf{V} , we know that $\{v_i\}_{i=1}^N$ are eigenvalues of $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ as $\mathbf{K}_{\mathbf{X},\mathbf{X}} = \mathbf{Q}\mathbf{V}\mathbf{Q}^T$.
 1029 Due to the definition of eigenvalue, there is a relation between matrix $\mathbf{K}_{\mathbf{X},\mathbf{X}}$,
 1030 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
 1031 $(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}) \mathbf{q}_i = v_i \mathbf{q}_i + \sigma_n^2 \mathbf{I} \mathbf{q}_i = (v_i + \sigma_n^2) \mathbf{q}_i$. Therefore, it can be de-
 1032 rived that if v_i is an eigenvalue of $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ then $v_i + \sigma_n^2$ is also an eigenvalue

1033 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
 1034 reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}\left(DN^{\frac{3}{D}}\right)$ which are the cost for eigendecomposition
 1035 of D matrices $\{\mathbf{K}_d(\mathbf{X}_d, \mathbf{X}_d)\}_{d=1}^D$.

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

1036 **Structural Kernel Interpolation.** Despite their impressive computational
 1037 acceleration of Kronecker-based methodology presented above, the main lim-
 1038 itation of this approach is the restriction of grid-structured data. However,
 1039 most datasets will not satisfy this requirement, making the application of such
 1040 techniques narrow. In order to relax the condition of having observations at
 1041 all possible input locations in the grid, there are several attempts such that
 1042 missing observations and incomplete grid are also permitted (Flaxman et al.,
 1043 2015; Wilson et al., 2014). Ultimately, Wilson and Nickisch (2015) have ex-
 1044 tended the concept of the Kronecker method to a general scenario with the
 1045 proposal of Kernel Interpolation for Scalable Structured Gaussian Processes
 1046 (KISS-GP) . This method constrains that the set of inducing positions \mathbf{Z}
 1047 constructs a complete multidimensional grid. Consider D -dimensional prob-
 1048 lems and introduce \mathbf{Z}_d as a vector containing distinct inducing locations along
 1049 with dimension d , we again define \mathbf{Z} as Cartesian product of $\mathbf{Z}_1, \dots, \mathbf{Z}_D$, i.e.
 1050 $\mathbf{Z} = \mathbf{Z}_1 \otimes \dots \otimes \mathbf{Z}_D$.

1051 Similarly, the size of \mathbf{Z} is $M = \prod_{d=1}^D G_d$ where G_d is the number of elements
 1052 in the vector \mathbf{Z}_d . By utilizing the tensor product kernel defined above, the
 1053 Kronecker idea enables the fast algebraic operations of the covariance matrix
 1054 of inducing points. Nevertheless, setting a massive M could be problematic
 1055 due to the time-consuming operations associated with the cross-covariance
 1056 $\mathbf{K}_{\mathbf{X},\mathbf{Z}}$ between design matrix \mathbf{X} and inducing locations \mathbf{Z} . For example, the
 1057 full covariance matrix $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ which can be expressed by Nyström approxima-
 1058 tion $\mathbf{K}_{\mathbf{X},\mathbf{Z}}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{X}}$ dominate the computations with quadratic complexity
 1059 to M , i.e. $\mathcal{O}(NM^2)$. Instead of computing directly $\mathbf{K}_{\mathbf{X},\mathbf{Z}}$, it is estimated
 1060 by interpolating on the $M \times M$ covariance matrix $\mathbf{K}_{\mathbf{Z},\mathbf{Z}}$. For example, if we
 1061 would like to estimate $k(\mathbf{x}, \mathbf{z}_j)$, for point \mathbf{x} and inducing input \mathbf{z}_j , we can start
 1062 by finding the two inducing points \mathbf{z}_a and \mathbf{z}_b which are the two closest to \mathbf{x} .
 1063 Then, we can estimate $k(\mathbf{x}, \mathbf{z}_j)$ by $\tilde{k}(\mathbf{x}, \mathbf{z}_j) = wk(\mathbf{z}_j, \mathbf{z}_a) + (1-w)k(\mathbf{z}_j, \mathbf{z}_b)$,
 1064 where w and $1-w$ are represented the relative distance from \mathbf{x} to \mathbf{z}_a and \mathbf{z}_b .
 1065 Generally, the cross covariance $\mathbf{K}_{\mathbf{X},\mathbf{Z}}$ between design matrix \mathbf{X} and inducing
 1066 points \mathbf{Z} can be interpolated by:

$$\mathbf{K}_{\mathbf{X},\mathbf{Z}} \approx \tilde{\mathbf{K}}_{\mathbf{X},\mathbf{Z}} = \mathbf{W}\mathbf{K}_{\mathbf{Z},\mathbf{Z}} \quad (2.88)$$

1068 While M is expected very large in the scenario, \mathbf{W} is constraint to be ex-
 1069 tremely sparse. There are several options to construct matrix \mathbf{W} based upon
 1070 various strategy including (i) local linear interpolation where each row of \mathbf{W}
 1071 contains only 2 non-zero entries or (ii) local cubic interpolation for greater
 1072 accuracy with 4 non-zero elements per row.

1073

1074 As a consequence, from the Nyström approximation to full covariance $\mathbf{K}_{\mathbf{X},\mathbf{X}}$, a
 1075 further estimation can be obtained by substituting $\tilde{\mathbf{K}}_{\mathbf{X},\mathbf{Z}}$ for $\mathbf{K}_{\mathbf{X},\mathbf{Z}}$. This gen-
 1076 eral approach of approximation is so called Structured Kernel Interpolation
 1077 (SKI) .

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

1078 By exploiting the fast Kronecker matrix-vector multiplications mentioned
 1079 above, the overall complexity of learning GP is $\mathcal{O}\left(DM^{1+\frac{1}{D}}\right)$ computations
 1080 and $\mathcal{O}\left(N + DM^{\frac{2}{D}}\right)$ storage. Nonetheless, this approach also introduces ad-
 1081 ditional design choices, such as determining the optimal density of the in-
 1082 terpolation point grid, which require further fine-tuning than the relatively
 1083 more straightforward inducing point methods. In general, the grid density is
 1084 expected to be heavily dependent on the choice of the kernel since more expres-
 1085 sive kernels are likely to require a greater number of interpolation points and
 1086 less sparse \mathbf{W} . In summary, the combination of SKI and Kronecker algebraic
 1087 structure results in the method KISS-GP.

1088 2.4 Random Feature Approximations

1089 As highlighted earlier, the inducing point-based approximation is a well-known
 1090 approach for improving GPs' scalability. In these methods, a small number of
 1091 pairs of inducing inputs and outputs are learned to define a new GPs, which
 1092 is expected to be close as possible to the GPs, and the computational and
 1093 storage cost now depend on the number of inducing points. These approaches
 1094 are appropriate for locally complex functions. Intuitively, most inducing in-
 1095 puts would be located in regions where the function is complex, while the
 1096 rest would be placed in regions where the function is simpler. Highly complex
 1097 functions cannot be modeled well with these inducing point-based approaches.

1098

1099 In order to capture complex behaviors at a global level and improve the
 1100 scalability of GPs, random feature-based approximations were proposed by
 1101 Lázaro-Gredilla et al. (2010) and Gal and Turner (2015), which relies on spec-
 1102 tral representations of kernel functions. For this kind of approximation, we

1103 only consider stationary GPs whose covariance functions are written as a func-
 1104 tion of the distance between observations, i.e. $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}') = k(\mathbf{r})$.
 1105 The spectral density for non-stationary kernels can be found in Remes et al.
 1106 (2017). The concept of spectral expressiveness and random feature expansions
 1107 are discussed here because these concepts are essential in the next chapter 2
 1108 where we propose a novel combination of CNNs and GPs approximated with
 1109 random features.

1110 2.4.1 Spectral Representations

1111 **Kernel trick and its problem.** Kernel methods are a class of algorithms
 1112 enabling the operations in an infinite-dimensional feature space, which leads
 1113 to an enhancement of representational power. This is materialized by ob-
 1114 serving that inference for these methods is expressed through inner products
 1115 between test points and input points, e.g. SVM (Cortes and Vapnik, 1995).
 1116 Thanks to this observations and Mercer’s theorem, we can implicitly define
 1117 the transformation from the original space to the infinite-dimensional space
 1118 by specifying the kernel function between points. This is the so-called *kernel*
 1119 *trick*. However, the weakness of these methods is that algorithms needs to
 1120 evaluate the kernel function between all pairs of datapoints. Consequently,
 1121 large training sets incur large computational and storage costs.

1122
 1123 **Dual representation of a stationary kernel.** Rahimi and Recht (2008)
 1124 proposed an idea to define a transformation of the input space enabling a
 1125 numerical approximation to kernel values without suffering a prohibitive cost.
 1126 Due to the significant impact on research communities working on kernel-
 1127 based models such as support vector machines, kernel ridge regression, and
 1128 ultimately GPs, this seminal work is considered to be one of the most in-
 1129 fluential papers published in the previous decade. Their work is inspired
 1130 by Bochner’s theorem (Rudin, 1962) which states that any continuous shift-
 1131 invariant normalized covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i - \mathbf{x}_j)$ is said to be
 1132 positive definite if and only if it can be rewritten as the Fourier transform
 1133 of some non-negative measure $p(\boldsymbol{\omega})$. The spectral density $s(\boldsymbol{\omega})$ can be con-
 1134 structed from $k(r)$ and vice versa through Wiener-Khintchin theorem:

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

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

1136 where \mathcal{F} denotes Fourier transform and $e^{ix} = \cos x + i \sin x$ is the Euler’s
 1137 formula. Thanks to the relation indicated in Equation 2.91 and 2.90, several

1138 examples of pairs of kernel function $k(r)$ and spectral density $p(\omega)$ can be
 1139 given as follows:

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

1139

1140

1141 **Approximation of RBF Kernel using Random Fourier Features.** Gen-
 1142 erally, we consider the RBF kernel parameterized by $\boldsymbol{\theta} = (\sigma^2, l_1, \dots, l_D)$ and its
 1143 corresponding spectral density can be found using Equation 2.91 as follows:

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

1144

$$p_{\text{RBF}}(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega} | \mathbf{0}, \boldsymbol{\Lambda}^{-1}), \text{ where } \boldsymbol{\Lambda} = \text{Diag}(l_1, \dots, l_D). \quad (2.93)$$

1145 From equation 2.90, the kernel function can be rewritten as the expectation
 1146 under the density $p_{\text{RBF}}(\boldsymbol{\omega})$.

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

1147 As $\sin(\cdot)$ is an odd function, i.e. $\sin(-x) = -\sin(x)$, the imaginary term can
 1148 be canceled out from the expectation in Equation 2.94. Further, the kernel
 1149 function can be approximated using N_{RF} spectral samples $\tilde{\boldsymbol{\omega}}$ from density
 1150 function $p(\boldsymbol{\omega})$.

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

1151 Replacing \mathbf{r} by $\mathbf{x}_i - \mathbf{x}_j$ into equation 2.95, we can express the approximation
 1152 of kernel function by an inner product representation:

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

1153 where $\phi_{\text{RBF}}(\mathbf{x})$ is known as random features of \mathbf{x} for RBF kernel, which is
1154 defined as follows:

$$\phi_{\text{RBF}}(\mathbf{x}) = \frac{\sigma^2}{N_{\text{RF}}} \left[\begin{array}{c} \cos(\mathbf{x}^T \tilde{\boldsymbol{\omega}}^{(1)}), \dots, \cos(\mathbf{x}^T \tilde{\boldsymbol{\omega}}^{(N_{\text{RF}})}) \\ \sin(\mathbf{x}^T \tilde{\boldsymbol{\omega}}^{(1)}), \dots, \sin(\mathbf{x}^T \tilde{\boldsymbol{\omega}}^{(N_{\text{RF}})}) \end{array} \right]^T \quad (2.97)$$

1155 **Approximation of order-one ARC-COSINE Kernel using Random Fea-**
1156 **tures.** In addition to working with RBF, we also consider order-one ARC-
1157 COSINE covariance which is a prevalent kernel function.

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

1158 where $\boldsymbol{\theta} = (\sigma, \Lambda = \text{Diag}(l_1^2, \dots, l_D^2))$ and α is the angle between $\Lambda^{-\frac{1}{2}} \mathbf{x}_i$ and
1159 $\Lambda^{-\frac{1}{2}} \mathbf{x}_j$. Let $H(\cdot)$ be the Heaviside function. Following Cho and Saul (2009),
1160 this covariance can be written under an integral form:

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

1161 The convenient integral representation allows for a Monte Carlo approxima-
1162 tion obtaining a low-rank approximation to the covariance matrix involving
1163 Rectified Linear Unit (ReLU) activation (Cho and Saul, 2009).

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

1164 2.4.2 Random featured-based Gaussian Processes.

1165 In this section, I firstly present a well-known study of approximation of GPs
1166 using random features, which is proposed by Lázaro-Gredilla et al. (2010).
1167 The key novel idea is to sparsify the spectral representation of GPs.

1168
1169 **Sparse Spectrum Gaussian Process Regression.** As alluded earlier,
1170 Gaussian Processes Regression (GPR) is introduced in function-space view.
1171 Here, we remind that, by considering the dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and Gaussian
1172 likelihood $p(\mathbf{y} | \mathbf{f}) = \mathcal{N}(\mathbf{y} | \mathbf{f}, \sigma_n^2 \mathbf{I})$, the predictive distributions $p(y_* | \mathbf{x}_*, \mathcal{D})$
1173 and the logarithm of the marginal likelihood $\log(\mathbf{y} | \boldsymbol{\theta})$ given parameters $\boldsymbol{\theta}$
1174 are expressed as follows:

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

1175

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

1176 Computing the gradients of logarithm of the marginal likelihood with respect
 1177 to related parameters requires the cubic cost to training size, i.e. $\mathcal{O}(N^3)$,
 1178 which is unacceptable for large-scale data sets. In order to avoid the pro-
 1179 hibitive cost, Lázaro-Gredilla et al. (2010) have employed the approximation
 1180 of the covariance matrix using spectral representation. Consider for example
 1181 ARD kernel (a stationary anisotropic squared exponential covariance function):

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

1182 where $\boldsymbol{\Lambda} = \text{Diag}([l_1^2, \dots, l_D^2])$. Based on the dual representation of the sta-
 1183 tionary kernel mentioned above, we can approximate the $k_{\text{ARD}}(\mathbf{x}_i, \mathbf{x}_j)$ using
 1184 N_{RF} spectral samples, and express the approximation as an inner product:

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

1185 where $\tilde{\boldsymbol{\omega}}^{(r)} \sim p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega} | 0, \boldsymbol{\Lambda}^{-1})$, and we define $\phi(\mathbf{x})$ as a column vector
 1186 of length $2N_{RF}$ containing the evaluation of the m pairs of trigonometric
 1187 functions at \mathbf{x} .

$$\phi(\mathbf{x}) = [\cos(\mathbf{x}^T \tilde{\boldsymbol{\omega}}^{(1)}), \dots, \cos(\mathbf{x}^T \tilde{\boldsymbol{\omega}}^{(N_{RF})}), \sin(\mathbf{x}^T \tilde{\boldsymbol{\omega}}^{(1)}), \dots, \sin(\mathbf{x}^T \tilde{\boldsymbol{\omega}}^{(N_{RF})})]^T \quad (2.105)$$

1188 From the transformation $\phi(\cdot)$, we construct $2N_{RF}$ by N matrix of random
 1189 features $\boldsymbol{\Phi}_{\mathbf{X}} = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N)]$. Now, the full kernel matrix $\mathbf{K}_{\mathbf{X}}$ can be
 1190 approximated as follows:

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

1191 Replacing the kernel matrix by this approximation in equation 2.101 and
 1192 2.102, we obtain the spectral approximation of predictive distribution with
 1193 mean μ_* and variance σ_*^2 :

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

1194 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
 1195 logarithm of the marginal likelihood:

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

1196 Since this method approximates kernel matrices using the spectral density, it
 1197 is called the Sparse Spectrum Gaussian Process (SSGP). Model selection can
 1198 be done by optimizing jointly the logarithm of the marginal likelihood defined
 1199 in 2.108 with respect to spectral points $\tilde{\omega}^{(r)}$ and hyperparameters θ . The
 1200 computational cost for each training step of SSGP algorithm is $\mathcal{O}(NN_{RF}^2)$. In
 1201 terms of making prediction for each test point, the cost is $\mathcal{O}(N_{RF})$ for the
 1202 predictive mean and $\mathcal{O}(N_{RF}^2)$ for the predictive variance.

1203
 1204 **Extensions of SSGP.** Gal and Turner (2015) show that the original SSGP
 1205 model’s have a tendency of overfitting. They have presented a Variational
 1206 Sparse Spectrum approximation to the Gaussian Processes (vSSGP) that al-
 1207 lows one to integrate out the set of spectral samples $\Omega = [\tilde{\omega}^{(1)}, \dots, \tilde{\omega}^{(N_{RF})}]$.
 1208 The model vSSGP is shown to yield better calibrated uncertainty estimates
 1209 accompanying predictions, and a procedure for deriving the optimal weights
 1210 analytically is given for the Gaussian likelihood case. Other approaches of
 1211 applying variational inference on SSGP are featured in Tan et al. (2015) and
 1212 Hoang et al. (2016). Besides, efficient random feature maps have also been
 1213 proposed to accelerate the computation and reduce the storage cost, such as
 1214 the Fastfood approximation (Le et al., 2013) and Orthogonal Random Fea-
 1215 tures (Yu et al., 2016).

1216 2.5 Local Approximation

1217 Inducing point-based and random feature-based approximations of GPs are
 1218 implemented based on a global distillation, and they are commonly used to
 1219 approximate GPs. However, these approaches require the computational and
 1220 storage costs which are determined by auxiliary variables, i.e. number of
 1221 inducing points or spectral samples. An alternative class of methods for im-
 1222 proving the scalability of GPs is to follow the divide-and-conquer idea, which
 1223 focuses on the local subsets of training data. According to the literature survey
 1224 conducted by (Liu et al., 2018b), in this text, we opt to split the approach of
 1225 local approximation into two groups: Separate-Local-Experts and Ensemble-
 1226 Local-Experts.

1227
 1228 **Separate-Local-Experts.** Intuitively speaking, there is almost no depen-
 1229 dence between two points which are distant from each other. Thus, the pre-
 1230 diction at an unseen input can be made sensibly by using localized experts with
 1231 an acceptable computational cost. For example, Kim et al. (2005) and Datta
 1232 et al. (2016) assume that a local expert model completely governs prediction
 1233 at inputs inside its corresponding area. Simply, these approaches firstly parti-

1234 tion the input space, then all local experts are trained based on these disjoint
 1235 subsets, and then the inference at \mathbf{x}_* can be made by an appropriate local
 1236 expert. By introducing \mathcal{M}_i as a local expert which is responsible for the sub-
 1237 region Ω_i and \mathcal{D}_i as the subset of data located inside Ω_i , we mathematically
 1238 state that the predictive distribution at \mathbf{x}_* can be approximated by using a
 1239 subset of data \mathcal{D}_i , i.e. $p(y_* | \mathcal{D}, \mathbf{x}_*) \approx p(y_i | \mathcal{M}_i, \mathcal{D}_i, \mathbf{x}_*)$. The partition on
 1240 input space can be made by some clustering algorithms, e.g. Voronoi tessella-
 1241 tions (Kim et al., 2005), and tree techniques (Vasudevan et al., 2009; Pratola
 1242 et al., 2013). By restricting the number of data points of a local model to M ,
 1243 there are N/M local GPs where N is training size. Learning all independent
 1244 GPs experts requires a cost of $\mathcal{O}(NM^2)$.

1245
 1246 Instead of grouping data points into disjoint subsets statically before training
 1247 local GPs experts, an alternative approach is to select a neighborhood subsets
 1248 \mathcal{D}_* around \mathbf{x}_* , and train a particular expert \mathcal{M}_* to make the prediction at
 1249 \mathbf{x}_* . For example, Urtasun and Darrell (2008) employ a dynamic partition to
 1250 choose m_0 neighbor points around \mathbf{x}_* , resulting in $\mathcal{O}(n_t m_0^3)$ complexity that
 1251 relies on the test size n_t . The primary problem of the approach is the concept
 1252 of the neighborhood set \mathcal{D}_* around \mathbf{x}_* . The most straightforward way is to
 1253 use geometric closeness criteria for selection, i.e. the selected points should be
 1254 close to \mathbf{x}_* . However, the approach is not optimal due to these closest points
 1255 convey redundant information. Thus, there are several GP-based methods
 1256 which have been employed to sequentially update the neighborhood set (Gra-
 1257 macy, 2016; Gramacy and Haaland, 2016; Gramacy and Lee, 2009; Gramacy
 1258 and Apley, 2015).

1259
 1260 While improving significantly the scalability and enjoying the capability of
 1261 capturing non-stationary features due to the localized structure, Separate-
 1262 Local-Experts yields discontinuous predictions on the boundaries of subre-
 1263 gions, which is illustrated in Liu et al. (2018b). To alleviate the discontinuity
 1264 problem, the patched GPs (Park and Huang, 2016; Park and Apley, 2018)
 1265 restricts that two adjacent local GPs are patched to share the nearly identical
 1266 predictions on the boundary. However, it possibly yields non-sensible predic-
 1267 tive variances, and are only available in low dimensional space (Pourhabib
 1268 et al., 2014b; Park and Apley, 2018). Another problem of Separable Local
 1269 Experts is to suffer from poor generalization since it misses the long-term
 1270 spatial correlations. To address the generalization issue, we can restrict that
 1271 all local expert use the same hyperparameters (Deisenroth and Ng, 2015), or
 1272 combine local and global approximation of GPs as mentioned in Snelson and
 1273 Ghahramani (2007).

1274

1275 **Ensemble-Local-Experts.** An alternative solution to mitigate the prob-
1276 lems raised by Separable Local Experts is to use the model averaging strategy,
1277 which is accomplished by an ensemble of local experts. The approach com-
1278 bines various local GPs possessing individual hyperparameters for enhancing
1279 accuracy and reliability (Yuksel et al., 2012a; Masoudnia and Ebrahimpour,
1280 2014). Mathematically, Ensemble-Local-Experts can be expressed as a mix-
1281 ture of M Gaussian model, where the weight for each component can be seen
1282 as a gating function of covariates, which often takes a parametric form such
1283 as the softmax (Jacobs et al., 1991) and probit function (Geweke and Keane,
1284 2007). More general, it can be extended to a tree-structured hierarchical ar-
1285 chitecture (Jordan and Jacobs, 1993).

1286

1287 The application of GPs mixture experts for big data scenarios must deal with
1288 various problems. For example, the question of determining the number of
1289 local experts can be dealt with by Akaike information criterion (Huang et al.,
1290 2014), or the synchronously balancing criterion (Zhao et al., 2015a). Another
1291 problem is on the reduction of computational cost, which includes several re-
1292 search directions. The first one is to the localization of experts. This can
1293 be accomplished by Expectation Maximization (ME) algorithm, wherein the
1294 data points are assigned to local experts through Maximum a Posterior in
1295 E-step (Nguyen and Bonilla, 2014b; Zhao et al., 2015b; Chen et al., 2014),
1296 and subsequently, the optimization in M-step only operates on small subsets
1297 of data. The second one is to combine global approximation with local ex-
1298 perts. When using m inducing points for each local GPs that is responsible
1299 for n samples, the complexity for training M experts is intuitively $\mathcal{O}(nm^2M)$,
1300 which can be reduced to $\mathcal{O}(nm^2)$ using hard-cut EM (Nguyen and Bonilla,
1301 2014b; Nguyen et al., 2016).

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,

1331 2005), which has shown that classifiers that use the standard cross-entropy
1332 loss are generally well calibrated. Perhaps surprisingly, modern CNNs, which
1333 are a particular case of deep neural networks (DNNs), have been found to be
1334 miscalibrated, and the depth of convolutional filters is the main factor affect-
1335 ing calibration (Guo et al., 2017). The work in Guo et al. (2017) shows that
1336 regularization, implemented through weight decay, improves calibration and
1337 that, ultimately, simple methods such as post-calibration (Platt, 1999) can be
1338 an effective remedy for most calibration issues of CNNs.

1339

1340 Alternatively, Bayesian CNNs (Gal and Ghahramani, 2016b) where convolu-
1341 tional filters are inferred using Bayesian inference techniques, seem like perfect
1342 candidates to model uncertainty in these architectures in a principled way.
1343 However, while Bayesian CNNs have been shown to be effective in obtaining
1344 state-of-the-art performance in image classification tasks, we are not aware of
1345 studies that show their calibration properties. Hence, our first contribution is
1346 to investigate the calibration properties of Bayesian CNNs.

1347

1348 Along a similar vein, independently of the works on Bayesian CNNs, there
1349 have been other attempts to give a probabilistic flavor to CNNs by combining
1350 them with Gaussian processes (GPs, (Rasmussen and Williams, 2006)). Most
1351 of these approaches can be seen as a way to parameterize a CNN-based covari-
1352 ance for GPs, and the aim is to learn end-to-end both the filters and the GPs
1353 (see, e.g., Bradshaw et al. (2017); Wilson et al. (2016)). A crucial aspect that
1354 the literature has overlooked, however, is that methods that combine CNNs
1355 and GPs suffer from the same issues of miscalibration that characterize mod-
1356 ern CNNs. Therefore, the second contribution of this paper is to show that
1357 current combinations of CNNs and GPs are miscalibrated.

1358

1359 Consequently, as our third contribution, we propose a novel combination of
1360 CNNs and GPs that is indeed well-calibrated, while being simple to imple-
1361 ment. In particular, we propose to replace the fully connected layers of CNNs
1362 with GPs that we approximate with random features (Cutajar et al., 2017;
1363 Lázaro-Gredilla et al., 2010). Due to this approximation, the resulting model
1364 becomes a Bayesian CNN with a nonlinear transformation applied to the con-
1365 volutional features. Building on the connection between variational inference
1366 and dropout, we apply Monte Carlo dropout (MCD, (Gal and Ghahramani,
1367 2016a)) to carry out joint inference over the filters and the approximate GPs,
1368 thus obtaining an end-to-end learning method for the proposed model, which
1369 we call CNN+GP(RF). The resulting approach is characterized by a number of
1370 attractive features: (i) it is well calibrated, given that it uses the multinomial
1371 likelihood and the filters are regularized using Bayesian inference techniques;

1372 (ii) it is as scalable as state-of-the-art CNNs, in so much as it can be trained
1373 using mini-batch updates and can exploit GPU and distributed computing;
1374 (iii) unlike other works that combine CNNs and GPs, it is as easy to implement
1375 as standard CNNs, as it leverages the equivalence of GPs approximated with
1376 random features and Bayesian DNNs (Cutajar et al., 2017; Gal and Turner,
1377 2015; Neal, 1996), and the connections between dropout and variational infer-
1378 ence (Gal and Ghahramani, 2016a). We extensively validate these properties
1379 in a variety of image classification tasks.

1380

1381 Our final contribution extends the above framework by replacing the last
1382 layer of CNNs with Deep GPs (Cutajar et al., 2017) and by proposing the use
1383 of structured random features to obtain faster and more compact GP approxi-
1384 mations (Le et al., 2013; Yu et al., 2016). In all, our proposal considerably im-
1385 proves on classification accuracy compared to previous combinations of CNNs
1386 and GPs (e.g., $\sim 88\%$ on CIFAR10 and $\sim 67\%$ on CIFAR100, all without data
1387 augmentation), while being competitive with state-of-the-art CNNs; we are not
1388 aware of other GP works that approach these results. Crucially, we achieve
1389 these performance without compromising on calibration, again considerably
1390 improving on previous approaches that combine CNNs and GPs.

1391 3.2 Related Work

1392 **Calibration of Convolutional Networks:** The issue of calibration of clas-
1393 sifiers in machine learning was popularized in the 90’s with the use of support
1394 vector machines for probabilistic classification (Platt, 1999). Calibration tech-
1395 niques aim to learn a transformation of the output using a validation set in
1396 order for the transformed output to give a reliable account of the actual prob-
1397 ability of class labels (Flach, 2016); interestingly, calibration can be applied
1398 regardless of the probabilistic nature of the untransformed output of the clas-
1399 sifier. Popular calibration techniques include Platt scaling (Platt, 1999) and
1400 isotonic regression (Zadrozny and Elkan, 2002).

1401

1402 Classifiers based on Deep Neural Networks (DNNs) have been shown to be
1403 well-calibrated (Niculescu-Mizil and Caruana, 2005). The reason is that the
1404 optimization of the cross-entropy loss promotes calibrated output. The same
1405 loss is used in Platt scaling and it corresponds to the correct multinomial like-
1406 lihood for class labels. Recent studies on the calibration of CNNs, which are
1407 a particular case of DNNs, however, show that depth has a negative impact
1408 on calibration, despite the use of a cross-entropy loss, and that regularization
1409 improves the calibration properties of classifiers (Guo et al., 2017).

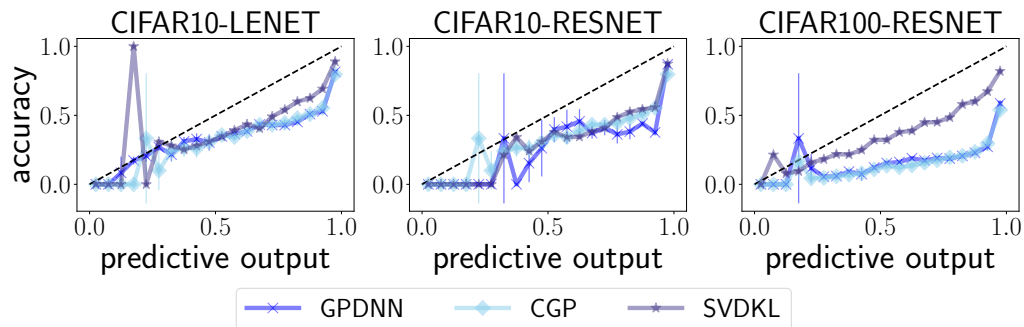


Figure 3.1 – Reliability diagrams for three state-of-the-art combinations of CNNs and GPs, i.e GPDNN (Bradshaw et al., 2017), CGP (van der Wilk et al., 2017), SVDKL (Wilson et al., 2016) applied to CIFAR10 and CIFAR100 data sets with LE_{NET} and RES_{NET} architectures. See table 3.1 for details on 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.

1410

1411 **Combinations of Conv Nets and Gaussian Processes:** Thinking of
 1412 Bayesian priors as a form of regularization, it is natural to assume that
 1413 Bayesian CNNs can “cure” the miscalibration of modern CNNs. Despite the
 1414 abundant literature on Bayesian DNNs (Neal, 1996; Mackay, 1994), far less
 1415 attention has been devoted to Bayesian CNNs (Gal and Ghahramani, 2016a),
 1416 and the calibration properties of these approaches have not been investigated.

1417

1418 Several approaches have proposed the combination of CNNs and GPs as a
 1419 means to give a probabilistic character to CNNs. Most of these works are
 1420 based on ideas developed in the context of manifold GPs (Calandra et al.,
 1421 2016), where inputs are transformed using some parametric transformation.
 1422 In these works, the parametric transformation is based on convolutional lay-
 1423 ers, and scalability to large data is achieved through the use of ideas drawn
 1424 from the literature on scalable GPs, for example the Stochastic Variational
 1425 Deep Kernel Learning (SVDKL) approach in Wilson et al. (2016). In contrast,
 1426 the work on hybrid GPs and DNNs (GPDNN, (Bradshaw et al., 2017)) com-
 1427 bines CNNs and GPs using an inducing point approximation. Other recent
 1428 approaches that aim to introduce convolutions in the calculation of the co-
 1429 variance between images include the work in van der Wilk et al. (2017), which
 1430 proposes a way to construct covariances between domains/patches, mimicking
 1431 the computations in CNNs.

1432

1433 In this work, we propose an alternative way to combine CNNs and GPs, where

1434 GPs are approximated using random features expansions (Rahimi and Recht,
 1435 2008; Lázaro-Gredilla et al., 2010). The random feature expansion approxima-
 1436 tion amounts to replacing the original kernel matrix with a low-rank approxi-
 1437 mation, turning GPs into Bayesian linear models. Combining this with CNNs
 1438 leads to a particular form of Bayesian CNNs, much like GPs and DGPs are par-
 1439 ticular forms of Bayesian DNNs (Duvenaud et al., 2014; Gal and Ghahramani,
 1440 2016a; Neal, 1996). Inference in Bayesian CNNs is intractable and requires
 1441 some form of approximation. In this work, we draw on the interpretation of
 1442 dropout as variational inference, employing the so-called Monte Carlo Dropout
 1443 (MCD, (Gal and Ghahramani, 2016a)) to obtain a practical way of combining
 1444 CNNs and GPs.

1445 3.3 On calibration of Convolutional GPs

1446 Consider a Q -class image classification task where \mathbf{X} denotes a set of N images
 1447 $\mathbf{x}_i \in \mathbb{R}^{p_x \times p_y}$ ($1 \leq i \leq n$), and \mathbf{Y} is the matrix consisting of the correspond-
 1448 ing one-hot encoded labels \mathbf{y}_i stacked by row. We can use various metrics
 1449 to determine the quality of a classifier, and here we focus in particular on
 1450 calibration.

1451
 1452 Let $\mathbf{g}(\mathbf{x})$ be the output of a classifier for an input image \mathbf{x} . To compute
 1453 the calibration properties of a classifier, consider a partitioning of the test
 1454 set \mathbf{X}_* into disjoint sets $\{\mathbf{X}_1, \dots, \mathbf{X}_M\}$, such that each subset \mathbf{X}_m contains
 1455 the inputs yielding predictions in the range $(\frac{m-1}{M}, \frac{m}{M}]$. Hence, the confidence
 1456 associated with each subset \mathbf{X}_m is characterized by the midpoint of its corre-
 1457 sponding range, i.e. $\text{conf}(\mathbf{X}_m) = \frac{m-0.5}{M}$. Then, the accuracy $\text{acc}(\mathbf{X}_m)$ for each
 1458 subset can be evaluated as follows:

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

1459 where $\delta(x)$ is equal to one if $x = 0$, and zero otherwise.

1460
 1461 In what follows, we use reliability diagrams to assess calibration, where we
 1462 plot accuracy as a function of confidence for the subsets $\{\mathbf{X}_1, \dots, \mathbf{X}_M\}$. For a
 1463 perfectly calibrated classifier, we expect $\text{acc}(\mathbf{X}_m) = \text{conf}(\mathbf{X}_m)$ for all m , with
 1464 deviations implying that the class probabilities are either underestimated or
 1465 overestimated. A useful summary statistics that can be extracted from reli-
 1466 ability diagrams is the *Expected Calibration Error* (ECE), which is the average
 1467 of the absolute difference between accuracy and confidence *weighted* according

1468 to its size:

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

1469 Another metric that measures the accuracy in predicting class probabilities is
 1470 the BRIER score which takes into account the factors of calibration, resolution
 1471 and uncertainty (Murphy, 1973). It is defined as the squared distance between
 1472 labels and outputs averaged across classes and test points:

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

1473 In figure 3.1, we report the reliability diagrams of three state-of-the-art com-
 1474 binations of CNNs and GPs, i.e GPDNN approach in Bradshaw et al. (2017),
 1475 CGP in van der Wilk et al. (2017) and SVDKL in Wilson et al. (2016). These
 1476 approaches are applied to the CIFAR10 and CIFAR100 data sets with vari-
 1477 ous convolutional structures. Note that the lines for CGP in the sub-figure of
 1478 CIFAR10-LENET and CIFAR10-RESNET are identical because there is no equiv-
 1479 alent CNN architecture in CGP. All of reliability diagrams for these methods
 1480 and ours can be found in the supplemental material.

1481
 1482 The results indicate that current approaches that combine CNNs and GPs are
 1483 miscalibrated, with a tendency of being overconfident in predictions. This is
 1484 an important and perhaps surprising finding, because one of the motivations
 1485 to combine CNNs with GPs is to do better quantification of uncertainty com-
 1486 pared to plain CNNs. In the experiments section we report more extensively
 1487 on the calibration of these classifiers, as well as illustrating other performance
 1488 metrics. These considerations call for the study of better ways to combine
 1489 CNNs and GPs to recover calibration while attempting to improve on standard
 1490 metrics such as error rate and test log-likelihood. The next section illustrates
 1491 our proposal that achieves this goal.

1492 3.4 Proposed Method

1493 In the proposed model, the labels \mathbf{Y}_i are assumed to be conditionally in-
 1494 dependent given a set of corresponding latent variables \mathbf{F}_i , i.e. we consider
 1495 the likelihood $p(\mathbf{Y}|\mathbf{F}) = \prod_{i=1}^N p(\mathbf{Y}_i | \mathbf{F}_i)$, where the latent variables \mathbf{F} are
 1496 realizations of a set of Q functions $f_j(\mathbf{x})$ at the input images $\mathbf{x}_1, \dots, \mathbf{x}_n$, i.e.,
 1497 $(\mathbf{F})_{ij} = f_j(\mathbf{x}_i)$ for $j = 1, \dots, Q$. Each individual $p(\mathbf{Y}_i | \mathbf{F}_i)$ is multinomial
 1498 with probabilities obtained using a softmax transformation of the latent vari-
 1499 ables. In this work we focus on functions $f_j(\mathbf{x})$ that are modeled using GPs;

1500 note that extension to DGPs is actually easy to consider in our framework, as
 1501 we show in the experiments.

1502

1503 Due to the GP modeling assumption, the latent function values $\mathbf{F}_{.j}$ compris-
 1504 ing $(f_j(\mathbf{x}_1), \dots, f_j(\mathbf{x}_n))^\top$ are jointly Gaussian with $p(\mathbf{F}_{.j}|\mathbf{X}, \boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$,
 1505 where \mathbf{K} is the covariance matrix. The entries of the covariance matrix
 1506 $\mathbf{K} = \{k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta})\}_{i,j}$, are specified by a covariance (kernel) function k (with
 1507 hyperparameters $\boldsymbol{\theta}$) and this form is shared across output dimensions, al-
 1508 though this can be relaxed and allow for a different k for the Q outputs.

1509

1510 Instead of applying the GP modeling directly to the images, we propose to em-
 1511 ploy a transformation $\mathbf{c}(\mathbf{x}|\Psi)$ using convolutional layers, where Ψ denotes the
 1512 parameters of such layers. The vector-valued function $\mathbf{c}(\mathbf{x}|\Psi)$ is differentiable
 1513 as it implements a series of differentiable operations, such as convolutions and
 1514 pooling. This is one of the key successes of CNN models that allows for the
 1515 learning of their filters, which we exploit for the end-to-end learning of our
 1516 model.

1517

1518 Inference in this model requires being able to characterize the posterior over
 1519 all or a selected group of model parameters, but this posterior is analytically
 1520 intractable and thus computationally prohibitive (Rasmussen and Williams,
 1521 2006). In the remainder of this paper, we build on previous work on scalable
 1522 inference for GPs and DGPs with random features (Cutajar et al., 2017) to ob-
 1523 tain an approximation to the proposed model that can be learned end-to-end.

1524 3.4.1 Random Feature Expansions

1525 Naïve inference in GP models requires algebraic operations with \mathbf{K} that would
 1526 cost $\mathcal{O}(n^3)$ in time. Popular approaches to recover tractability use low-rank
 1527 approximations of the kernel matrix. Among this family of low-rank approx-
 1528 imations, we choose to work with random feature approximations (Lázaro-
 1529 Gredilla et al., 2010; Cutajar et al., 2017). The reason is that they offer
 1530 a number of possible extensions to speedup computations (e.g., using struc-
 1531 tured approximations (Le et al., 2013; Yu et al., 2016)) and increase the com-
 1532 plexity of the model (e.g., considering Deep GPs (Cutajar et al., 2017)); we
 1533 elaborate on this in the experiments section. In random feature expansions,
 1534 the kernel matrix is replaced by a low-rank approximation $\mathbf{K} \approx \Phi\Phi^\top$, with
 1535 $\Phi \in \mathbb{R}^{n \times m}$ and $m \ll n$. This approximation suggests the construction of a
 1536 Bayesian linear model to approximate the GP latent variables as $\mathbf{F} = \Phi\mathbf{W}$.
 1537 Using $p(W_{ij}) = \mathcal{N}(W_{ij}|0, 1)$ it is straightforward to show that the covari-
 1538 ance of each of the latent functions $\mathbf{F}_{.j}$ is indeed an approximation to \mathbf{K} , as

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

1540

1541 In this work, we focus in particular on the order-one ARC-COSINE kernel (Cho
1542 and Saul, 2009)

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

1543 where $\boldsymbol{\theta} = (\sigma, \mathbf{\Lambda} = \text{Diag}(\ell_1^2, \dots, \ell_d^2))$ and α is the angle between $\Lambda^{-\frac{1}{2}} \mathbf{c}(\mathbf{x}_i | \mathbf{\Psi})$
1544 and $\Lambda^{-\frac{1}{2}} \mathbf{c}(\mathbf{x}_j | \mathbf{\Psi})$.

1545

1546 The ARC-COSINE covariance has a convenient integral representation that al-
1547 lows for a Monte Carlo approximation, obtaining a low-rank approximation
1548 to the covariance matrix involving Rectified Linear Unit (ReLU) activations
1549 (Cho and Saul, 2009)

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

1550 In this expression, we have defined $\mathbf{C}(\mathbf{X} | \mathbf{\Psi})$ as the matrix resulting from
1551 the application of convolutional layers to the image training set \mathbf{X} and $\mathbf{\Omega}$
1552 is obtained by stacking N_{RF} samples from $p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega} | \mathbf{0}, \mathbf{\Lambda}^{-1})$ by column.
1553 Note that in the case of a popular Radial Basis Function (RBF) covariance,
1554 it is possible to obtain a similar random feature approximation, where the
1555 ReLU activation is replaced by trigonometric functions; see Rahimi and Recht
1556 (2008) and the supplement for details.

1557 3.4.2 End-to-end learning

1558 Inference in the proposed model is intractable due to the likelihood that is
1559 not conjugate to the GP prior. Further complications stem from the need to
1560 infer kernel parameters, which include convolutional parameters, and the need
1561 to be able to scale to large data. Our aim is to carry out inference within a
1562 consistent framework that is characterized by simplicity, as described next.

1563

1564 We start by introducing an approximate posterior over \mathbf{W} , $\mathbf{\Omega}$ and $\mathbf{\Psi}$, that we
1565 denote as $q(\mathbf{W}, \mathbf{\Omega}, \mathbf{\Psi})$. Following standard variational inference arguments,
1566 we can define an operative way to obtain these approximate posteriors. The
1567 log-marginal likelihood $\mathcal{L} = \log [p(\mathbf{Y} | \mathbf{X}, \boldsymbol{\theta})]$ can be bounded by the sum of an
1568 expected log-likelihood term and a negative Kullback-Leibler (KL) divergence

1569 term as follows:

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

1570 Variational inference amounts to optimizing the lower bound above with re-
1571 spect to $q(\mathbf{W}, \Omega, \Psi)$ and any other parameters of interest.

1572

1573 We have now a number of options on the form for the approximate poste-
1574 riors $q(\mathbf{W}, \Omega, \Psi)$. In previous works on variational inference for DNNs, it has
1575 been proposed to define the approximating distributions to be Gaussian and
1576 factorized across parameters (Kingma and Welling, 2014; Graves, 2011). The
1577 drawback of this is that it doubles the number of parameters. Alternatively,
1578 we can rely on the connections between dropout and variational inference
1579 (Gal and Ghahramani, 2016a,b) which is drawn by assuming the posterior of
1580 \mathbf{W}, Ω and Ψ as a mixture of two Gaussian distributions (see supplement).
1581 From this connection, we are able to obtain an easier approximate inference
1582 scheme, which is also known as Monte Carlo Dropout (MCD). Focusing on
1583 the weights for now, the connection with dropout is apparent if we rewrite

$$\mathbf{W} = \mathbf{M}_w \text{Diag}[\mathbf{z}_w] \quad (3.7)$$

with $(\mathbf{z}_w)_i \sim \text{Bernoulli}(\pi_w)$. 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 reparameterization can be done for the convolutional parameters Ψ and matrices of random feature Ω , 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

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

1584 The expectation in (3.16), instead, can be unbiasedly estimated using Monte
1585 Carlo and also considering a mini-batch of size m :

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

1586 with $\mathbf{W}^{(i)}, \Omega^{(i)}, \Psi^{(i)} \sim q(\mathbf{W}, \Omega, \Psi)$, and \mathcal{I}_m is a set of m indices to select
 1587 a mini-batch of training points (Graves, 2011). This doubly-stochastic ap-
 1588 proximation is differentiable wrt variational parameters when the Bernoulli
 1589 variables are fixed.

1590

1591 The approximate objective can now be optimized in the same vein as in stan-
 1592 dard back-propagation with dropout, noting that dropout is applied to \mathbf{W} ,
 1593 Ω and to convolutional parameters Ψ . What changes, however, is the in-
 1594 terpretation of the procedure as stochastic variational inference, whereby the
 1595 Bernoulli variables are resampled at each iteration. A practical implication is
 1596 in the way we compute the predictive distribution, which has a probabilistic
 1597 flavor as follows:

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

1598 and can be approximated using Monte Carlo by resampling the Bernoulli
 1599 variables. While MCD has been proposed for CNNs in (Gal and Ghahramani,
 1600 2016b), in this work we extend it to the case of joint inference over convolu-
 1601 tional parameters and the GP approximation in the CNN+GP(RF) model, thus
 1602 obtaining a practical inference and prediction scheme, which combines CNNs
 1603 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.

1604 3.4.3 Extensions

1605 **Structured random feature approximations:** One of the advantages of
 1606 the proposed model, compared to other GP approximations, is that it can
 1607 exploit structured random feature expansions to accelerate computations and
 1608 reduce the size of the approximate GP (Le et al., 2013; Yu et al., 2016). In the
 1609 random features approximation, random features are constructed by multiply-
 1610 ing Ω with the convolutional features. Without loss of generality, assuming
 1611 that $\Omega \in \mathbb{R}^{m \times d}$ and $\mathbf{c}(\mathbf{x}|\Psi) \in \mathbb{R}^{d \times 1}$, the cost of computing products $\Omega\mathbf{c}(\mathbf{x}|\Psi)$
 1612 is $\mathcal{O}(md)$, while storing Ω requires $\mathcal{O}(md)$ storage.

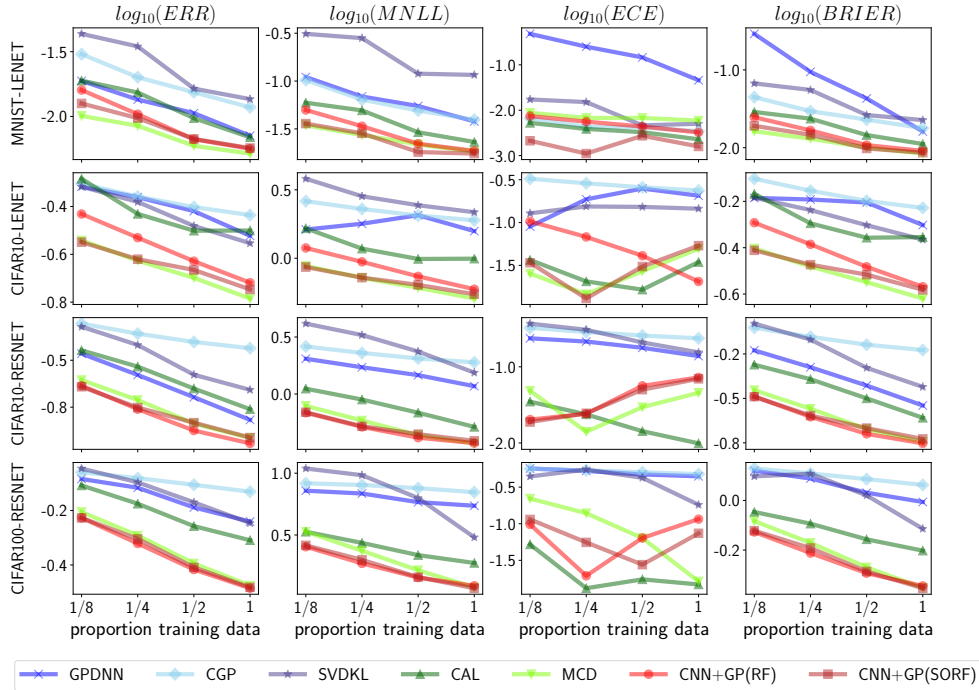


Figure 3.2 – Comparison of our CNN+GP(RF) and CNN+GP(SORF) with existing combinations of CNNs with GPs, and with Bayesian CNNs and post-calibrated CNNs. All performance metrics are defined so that the lower the better.

1613 Structured approximations aim to reduce the time complexity to $\mathcal{O}(m \log d)$
 1614 and the storage cost to $\mathcal{O}(m + d)$. Taking a standard random features expansion
 1615 of the isotropic covariance in (3.5) with $\mathbf{\Lambda} = \ell^{-2}\mathbf{I}$ as an example, $\mathbf{\Omega} = \frac{1}{\ell}\mathbf{G}$,
 1616 with $\mathbf{G}_{ij} \sim \mathcal{N}(0, 1)$. One way to make computations cheaper is to replace
 1617 the Gaussian matrix \mathbf{G} with a pseudo-random alternative. The Structured
 1618 Orthogonal Random Feature (SORF) approximation (Yu et al., 2016) approx-
 1619 imates \mathbf{G} through a series of Hadamard transformations of diagonal matrices
 1620 \mathbf{D}_i with elements randomly sampled from $\{-1, +1\}$ or Rademacher distri-
 1621 bution, 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-
 1622 Hadamard matrix. We refer to this variation of the model as CNN+GP(SORF).

1623 Similarly to the other parameters, we infer the diagonal matrices \mathbf{D}_i using
 1624 MCD. We denote by \mathbf{d}_i the diagonal of $\mathbf{D}_i, i = 1, 2, 3$. The MCD scheme
 1625 (Gal and Ghahramani, 2016a,b) assumes an L_2 regularization which implies
 1626 a zero-mean Gaussian prior, which is inappropriate for \mathbf{d}_i as it is Rademacher
 1627 distributed. We propose to bypass this limitation by applying MCD to a
 1628 reparameterization of \mathbf{d}_i . In particular, denoting by $\mathbf{d}_i^* \in \{-1, +1\}^d$ the
 1629 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 $\mathbf{M}_{\mathbf{d}_i - \mathbf{d}_i^*}$

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

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.

1665

1666 In TEMP it is necessary to leave out part of the data to perform post-calibration,
1667 which can be problematic in applications where obtaining labeled data is dif-
1668 ficult or expensive. As a result, our proposal is considerably better, although
1669 TEMP is competitive in ECE; this is expected given that this is the metric that
1670 is optimized after training.

1671

1672 The two variants of our approach, namely CNN+GP(RF) where we learn the
1673 frequencies Ω and CNN+GP(SORF) where we sample Ω from its prior, are
1674 comparable. This suggests that the extra level of complexity of learning the
1675 spectral frequencies does not lead to substantial gains in performance and that
1676 the structured random feature approximation yields satisfactory performance.

1677

1678 We also note that these results have been obtained by fixing the covariance
1679 parameters θ of the GP, as we found it to be unstable when learning these
1680 jointly with Ω . This might be the reason why these parameters were learned
1681 through cross-validation in Gal et al. (2017). In the supplement, we report
1682 the results obtained when learning θ and fixing Ω , which we found yielding
1683 similar performance as fixing θ . All these observations corroborate the hy-
1684 pothesis that most of the performance of CNN-based classification models is
1685 due to the convolutional layers.

1686

1687 In summary, figure 3.2 shows that our CNN+GP(RF) is the best strategy for
1688 calibrating these models compared to other approaches using GPs. Further-
1689 more, we found perhaps surprisingly that MCD has comparable performance.
1690 In the supplementary material, we report results on GPDNN where we infer
1691 convolutional parameters using MCD, so as to gain insights as to whether
1692 most of the improvements in performance are due to this form of regulariza-
1693 tion. The results support the intuition that inferring these parameters yields
1694 improvements in calibration, but also that our CNN+GP(RF) still offers better
1695 performance.

1696 3.5.1 Reliability diagrams

1697 In figure 3.3, we report the reliability diagrams of all the methods studied in
1698 figure 3.1. The figure shows that TEMP, MCD and CNN+GP(RF) produce well-
1699 calibrated predictions when using a shallow convolutional structure (LENET).
1700 For a deeper architecture (RESNET), CNN+GP(RF) is slightly under-confident.
1701 Compared to previous combinations of CNNs and GPs, our approach yields
1702 better reliability curves.

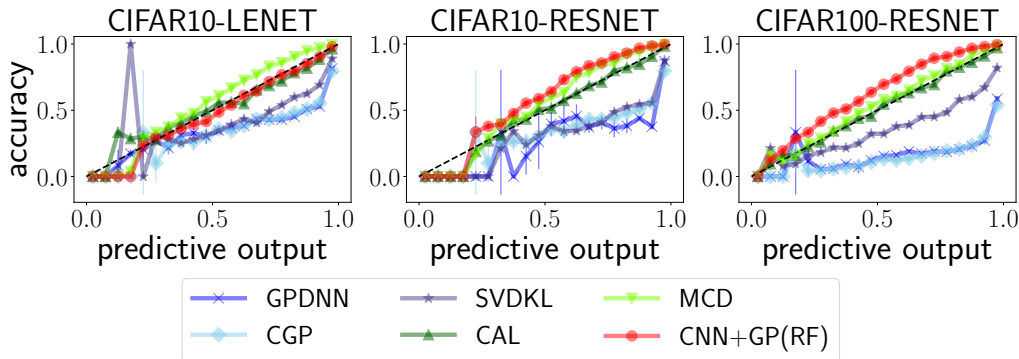


Figure 3.3 – Reliability diagrams of our CNN+GP(RF) in comparison with existing combinations of CNNs with GPs, and with Bayesian CNNs and post-calibrated CNNs.

1703 3.5.2 Extension with Deep GPs

1704 In figure 3.4, we report results varying the depth of a DGP on top of the
 1705 convolutional layers; again, we learn the convolutional filters and the DGP
 1706 end-to-end as discussed in the previous sections. We show results when ap-
 1707 plying our model to the whole CIFAR10 data set in the case of the shallow
 1708 convolutional structure (table 3.1). We feed-forward the convolutional fea-
 1709 tures to all layers of the DGP, in line with what suggested in the literature
 1710 of DGPs to avoid pathologies in the functions that can be modeled (Cutajar
 1711 et al., 2017; Duvenaud et al., 2014; Neal, 1996). The results indicate that
 1712 increasing the complexity of the model improves on all performance metrics,
 1713 and worsen calibration, which however is still around 3% ECE. This is in
 1714 line with the intuition that increasing model complexity negatively impacts
 1715 calibration.

1716 3.5.3 Knowing when the model does not know

1717 We report experiments showing the ability of our model to know when it does
 1718 not know, following a similar experimental setup as in Lakshminarayanan
 1719 et al. (2017). In this experiment we train our CNN+GP(RF) model on MNIST
 1720 and test on the NOT-MNIST dataset, which contains images of letters from
 1721 “A” to “J” in various typefaces. For this experiment, while we do not know
 1722 the exact value that we should obtain for predictive probabilities, we expect
 1723 to observe low entropy in the predictions when testing on MNIST and high
 1724 entropy when predicting on NOT-MNIST, indicating high uncertainty. The re-
 1725 sults are reported in figure 3.5, where we show the cumulative distribution
 1726 of the entropy of predictive probabilities for two depths of the convolutional
 1727 structure. In the figure, we compare our CNN+GP(RF) against one of the

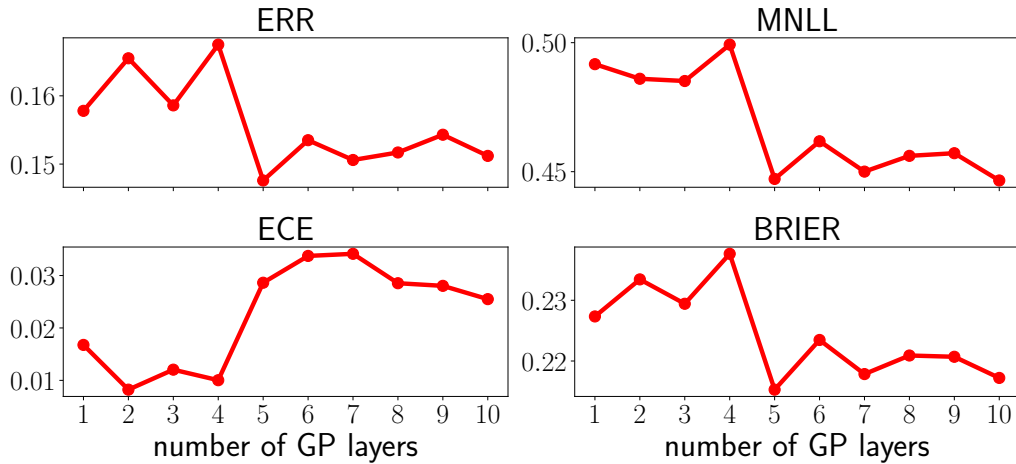


Figure 3.4 – Performance of the proposed model when varying the depth of the DGP on top of a RESNET convolutional structure on CIFAR10 dataset. Note that the scale of y -axes indicates that the metrics change only slightly when increasing the depth of the DGP.

1728 methods combining CNNs and GPs, that is GPDNN. In the figure, we also include
 1729 results on CNNs with post-calibration and Bayesian CNNs inferred with
 1730 MCD. Our approach is competitive with Bayesian CNNs and it is considerably
 1731 superior to post-calibration. This is especially true in the case of the RESNET
 1732 convolutional structure, where post-calibration still yields a large number of
 1733 predictions with low uncertainty. Interestingly, GPDNN assigns large uncertainty
 1734 to predictions on NOT-MNIST, although with the deeper convolutional
 1735 architecture it yields a large fraction of predictions with low entropy. We
 1736 speculate that this is due to the inducing point approximation of the GP, which
 1737 nicely captures uncertainty away from training data except for test points
 1738 which are closer to the training data.

1739 3.5.4 Extension with the SORF

1740 In table 3.2, we report further results comparing MCD with CNN+GP(SORF).
 1741 In this experiment, we use the ALEXNET structure (Krizhevsky et al., 2012) on
 1742 CIFAR10 and CIFAR100 datasets. The results in table 3.2 show improvements
 1743 in using our model compared to CNNs with MCD. We attribute this to the fact
 1744 that the GP approximated through SORF in place of the fully connected layer
 1745 of ALEXNET reduces model parameters from 30 million to 2.3 million.

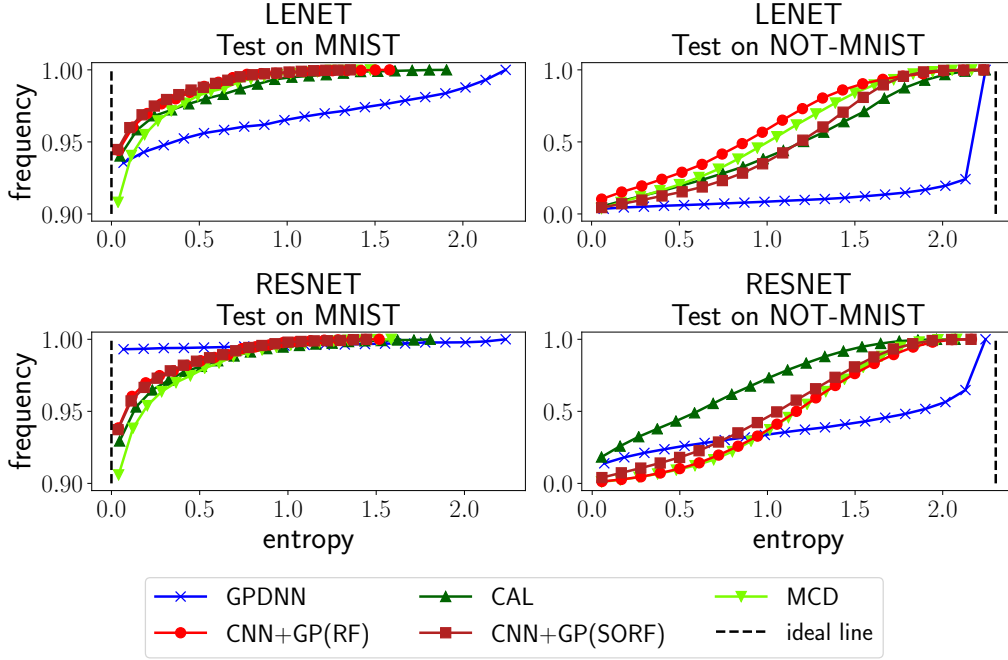


Figure 3.5 – Cumulative distribution function plot of predictive entropies when the models trained on MNIST are tested on MNIST and NOT-MNIST. We report results for two different depths of the convolutional structure. NOT-MNIST dataset available at <http://yaroslavvb.blogspot.fr/2011/09/notmnist-dataset.html>

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.

1746 3.6 Mathematical details and other experiments

1747 3.6.1 Random Feature Expansion of the RBF Covariance

1748 We report here the expansion of the popular Radial Basis Function (RBF) co-
 1749 variance. Following the convolutional representation of images in our CNN+GP(RF)
 1750 model, the RBF covariance is defined as:

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

1751 with $\boldsymbol{\theta} = (\sigma, \boldsymbol{\Lambda} = \text{Diag}(\ell_1^2, \dots, \ell_d^2))$. It is possible to express this covariance
 1752 function as the Fourier transform of a non-negative measure $p(\boldsymbol{\omega})$ Rahimi and
 1753 Recht (2008), where $\boldsymbol{\omega}$ are the so-called spectral frequencies. It is straightfor-
 1754 ward to verify that $p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}|\mathbf{0}, \boldsymbol{\Lambda}^{-1})$. Stacking N_{RF} Monte Carlo samples
 1755 from $p(\boldsymbol{\omega})$ into $\boldsymbol{\Omega}$ by column, we obtain

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

1756 where $\mathbf{C}(\mathbf{X}|\boldsymbol{\Psi})$ denotes the matrix resulting from the application of convolu-
 1757 tional layers to the image training set \mathbf{X} , and the sin and cos functions are
 1758 applied elementwise to their argument.

1759 3.6.2 Variational Inference for the Proposed Model

1760 3.6.2.1 CNN+GP(RF)

1761 In CNN+GP(RF), the variational parameters we would like to optimize are
 1762 $\mathbf{M}_w, \mathbf{M}_\psi$ and \mathbf{M}_Ω . Our model parameters $\mathbf{W}, \boldsymbol{\Psi}$ and $\boldsymbol{\Omega}$ share an identical
 1763 form for the approximate posterior and prior. Focusing on \mathbf{W} , its elements
 1764 have a standard normal prior, and we assume that the posterior $q(\mathbf{W})$ is
 1765 a mixture of two Gaussian distribution, which can be factorized over rows,
 1766 governed by variational parameters \mathbf{M}_w :

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

1767 where $\pi_w \in [0, 1]$, $\sigma^2 \approx 0$ and $\mathbf{M}_{w_r} \in \mathbb{R}^D$. This form of posterior leads
 1768 to the sampling procedure which characterizes dropout Gal and Ghahramani
 1769 (2016a,b). Given the choice of $\sigma^2 \approx 0$, \mathbf{W} can be sampled by introducing
 1770 Bernoulli variables

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

1771 and similarly for $\boldsymbol{\Psi}$ and $\boldsymbol{\Omega}$.

1772 All variational parameters are optimized to maximize the lower bound of
 1773 marginal likelihood which is defined as follows

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

1774 The expectation in 3.16 can be unbiasedly estimated using Monte Carlo and
 1775 also considering a mini-batch of size m

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

1776 where $\mathbf{W}^{(i)}, \Psi^{(i)}, \Omega^{(i)}$ is a sample from $q(\mathbf{W}, \Psi, \Omega)$, and can be obtained via
 1777 3.15. \mathcal{I}_m is a set of m indices to select a mini-batch of training points. In
 1778 classification, each individual $p \left(\mathbf{y}_k | \mathbf{x}_k, \mathbf{W}^{(i)}, \Psi^{(i)}, \Omega^{(i)}, \theta \right)$ can be computed
 1779 using a softmax transformation. The KL term can be approximated following
 1780 Gal and Ghahramani (2016a), noting that the fact that we are treating Ω
 1781 variationally, gives rise to extra terms that involve the GP length-scale ℓ :

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

1782 3.6.2.2 CNN+GP(SORF)

1783 In CNN+GP(SORF), our proposed variational inference scheme is similar to the
 1784 one in CNN+GP(RF), except that Ω is replaced by $l^{-1} \sqrt{N_{\text{RF}}} \mathbf{H} \mathbf{D}_1 \mathbf{H} \mathbf{D}_2 \mathbf{H} \mathbf{D}_3$,
 1785 with length-scale l and $\mathbf{D}_i = \text{Diag}(\mathbf{d}_i)$ and \mathbf{H} is the normalized Walsh-
 1786 Hadamard matrix. Because \mathbf{d}_i is Rademacher distributed, the form of prior
 1787 and posterior in MCD proposed by Gal and Ghahramani (2016a,b) is inade-
 1788 quate. Therefore, we use the prior $p_\varepsilon(\mathbf{d}_i) = \mathcal{N}(\mathbf{d}_i | \mathbf{d}_i^*, \varepsilon^2 \mathbf{I}_{N_{\text{RF}}})$ with \mathbf{d}_i^* sampled
 1789 from the Rademacher distribution and a small positive ε . The posterior $q(\mathbf{d}_i)$
 1790 is also composed by two Gaussian distribution as in CNN+GP(RF)

$$q(\mathbf{d}_i) = \prod_{j=1}^{N_{\text{RF}}} q([\mathbf{d}_i]_j) \quad (3.19)$$

$$, \text{ where } q([\mathbf{d}_i]_j) = \pi_d \mathcal{N}(\mathbf{M}_{[\mathbf{d}_i]_j}, \sigma^2) + (1 - \pi_d) \mathcal{N}([\mathbf{d}_i^*]_j, \sigma^2)$$

1791 with $\pi_d \in [0, 1], \sigma^2 \approx 0$ and $\mathbf{M}_{\mathbf{d}_i} \in \mathbb{R}^{N_{\text{RF}}}$. Following Gal and Ghahramani
 1792 (2016a), we can approximate the KL term between $q(\mathbf{d}_i)$ and $p(\mathbf{d}_i)$

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

1793 In terms of implementation, we do not apply MCD to $\mathbf{d}_i - \mathbf{d}_i^*$ but on \mathbf{d}_i directly.
 1794 According to this choice, each element in \mathbf{d}_i is sampled based on the variational

1795 parameters $M_{\mathbf{d}_i - \mathbf{d}_i^*}$ as in 3.21. Thanks to this trick, the implementation of
 1796 MCD scheme does not change for optimizing \mathbf{d}_i

$$\mathbf{d}_i = \begin{cases} M_{\mathbf{d}_i - \mathbf{d}_i^*} + \mathbf{d}_i^*, & \text{with probability } \pi_d \\ \mathbf{d}_i^*, & \text{otherwise} \end{cases} \quad (3.21)$$

1797 In figure 3.6, we report some experimental results to illustrate the impact of
 1798 optimizing \mathbf{d}_i . For CIFAR10-LENET and CIFAR100-RESNET, the optimization
 1799 of SORF parameters outperforms the case where spectral frequencies are fixed
 1800 in terms of ERR, MNLL and BRIER. In the case of CIFAR10-RESNET, the gains
 1801 are marginal.

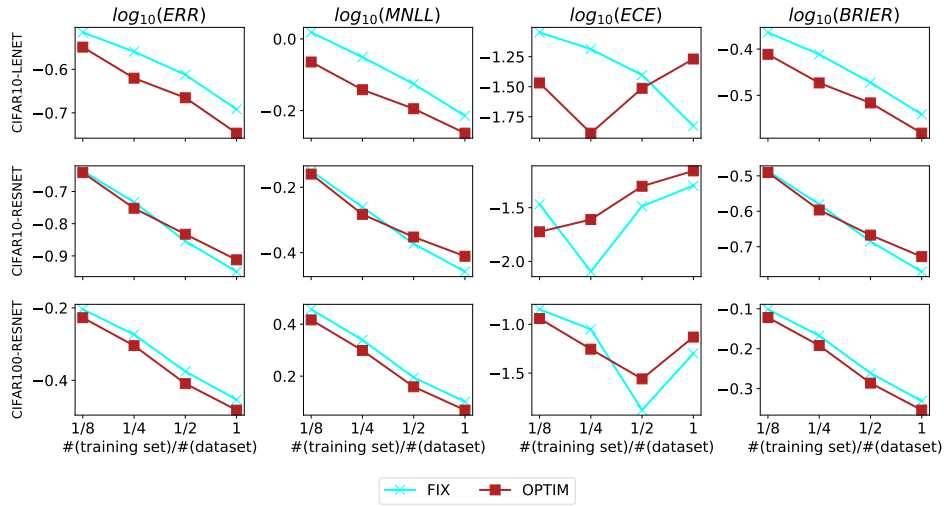


Figure 3.6 – Impact of optimization of SORF parameters

1802 **3.6.2.3 Optimization for covariance parameters**

1803 When using 3.18 to optimize all variational parameters pertaining to $q(\mathbf{W}, \Psi, \Omega)$
 1804 jointly with covariance θ we encountered some instabilities, and therefore we
 1805 decided to report results when fixing the covariance parameters θ in our pa-
 1806 per. For the case where Ω is not learned variationally we can simply draw Ω
 1807 from the prior $\mathcal{N}(\Omega_j | \mathbf{0}, \Lambda^{-1})$ and consider the reparameterization:

$$\Omega_{.j} = \Lambda^{-\frac{1}{2}} \varepsilon, \quad (3.22)$$

1808 where $\varepsilon_i \sim \mathcal{N}(\varepsilon_i | 0, 1)$ (Lázaro-Gredilla et al., 2010). This reparameterization
 1809 allows for the update of covariance parameters θ fixing the randomness in the
 1810 sampling from $p(\Omega | \theta)$. The results comparing CNN+GP(SORF) when updating
 1811 or fixing θ throughout optimization are reported in table 3.3. It is interesting
 1812 to notice how fixing covariance parameters θ leads to comparable performance
 1813 to the case where they are learned.

Table 3.3 – Results on the proposed CNN+GP(SORF) when fixing or learning covariance parameters θ . 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

1814 3.6.3 Variational inference of filters in GPDNN

1815 In this section we report results when applying variational inference on the
 1816 weights in GPDNN (Bradshaw et al., 2017). In order to do this, we implemented
 1817 MCD for the convolutional parameters, similarly to what presented in the main
 1818 paper for our CNN+GP(RF) model. The results in table 3.4 indicate that this
 1819 improves the calibration and accuracy of GPDNN compared to optimizing the
 1820 filters. In the case of a shallow convolutional architecture, the performance
 1821 of CNN+GP(RF) and GPDNN are comparable, although in the deeper case
 1822 CNN+GP(RF) achieves better performance. This supports the intuition that
 1823 inferring convolutional parameters, rather than optimizing them, leads to
 1824 considerable improvements in calibration.

1825 3.6.4 Reliability diagrams

1826 In this section, we report the reliability diagram and histogram of predictive
 1827 output for all methods with various datasets, i.e CIFAR10 and CIFAR100 and
 1828 convolutional architectures, i.e LENET and RESNET. We use the best config-
 1829 uration for CGP according to the implementation released by the Authors. In
 1830 each figure, rows correspond with the dataset and convolutional architecture,
 1831 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

1832 els are evaluated on the entire testing set. The number of bins used to draw
 1833 the reliability diagram is 20.

1834

1835 In each subfigure, the dashed line indicates perfect calibration. The horizontal
 1836 axis is the softmax output ranging from 0 to 1. The vertical axis indicates
 1837 accuracy rate for the red line or frequency for the green bars. The red dot
 1838 is the real average accuracy at each bin, while the line segments at the red
 1839 dots refer to the standard deviation of the accuracies. The green bar is the
 1840 average frequency histogram at each bin of softmax values. The experiments
 1841 of GPDNN, CGP, MCD-CIFAR10-LENET and CNN+GP(RF) are repeated three
 1842 times.

1843

1844 Having observed these figures, we see that regularizing convolutional filters
 1845 has a huge impact on calibration. From figures 3.7, 3.8, 3.9 and 3.10 we see
 1846 that CNNs and the previous combinations of GPs and CNNs are miscalibrated.
 1847 From figure 3.12 and 3.13, instead, we see that Bayesian CNNs improve the
 1848 reliability of the prediction, which is comparable with post-calibration.

1849

1850 It seems that there is a correlation between the histogram of predictive out-
 1851 put and the reliability line. When the histogram is skewed to the right, the
 1852 corresponding classifier is poorly calibrated.

1853

1854 Post calibration, MCD and CNN+GP(RF) (our method) are able to yield cali-
 1855 brated classification.

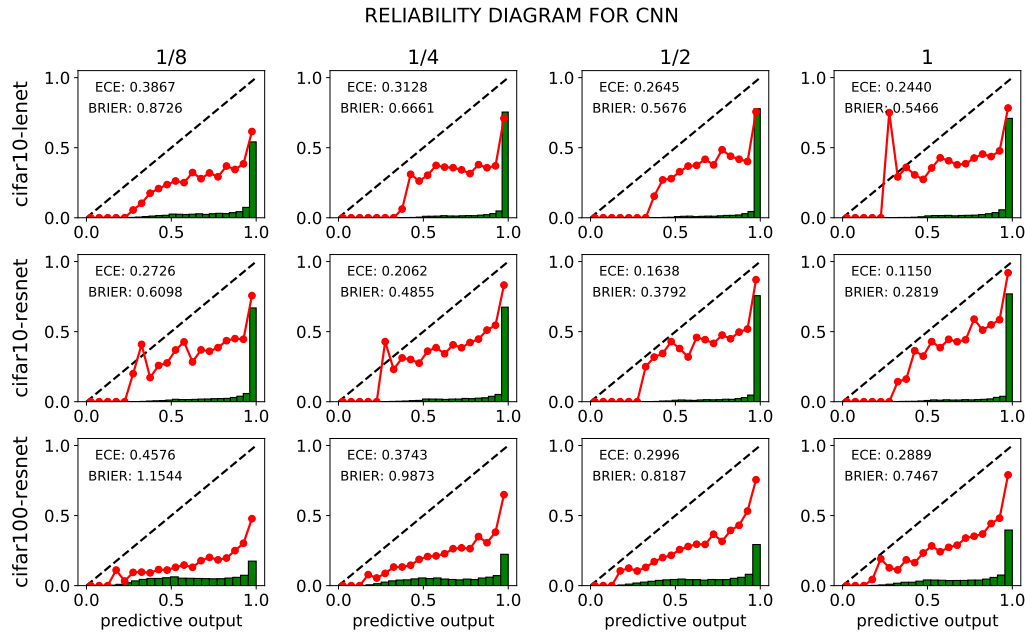


Figure 3.7 – Reliability diagrams for CNN

1856 3.7 Conclusions

1857 Despite the considerable interest in combining CNNs with GPs, little attention
 1858 has been devoted to understand the implications in terms of the ability of these
 1859 models to accurately quantify the level of uncertainty in predictions. This is
 1860 the first work that highlights the issues of calibration of these models, showing
 1861 that GPs cannot cure the issues of miscalibration in CNNs. We have proposed
 1862 a novel combination of CNNs and GPs where the resulting model becomes
 1863 a particular form of a Bayesian CNN for which inference using variational
 1864 inference is straightforward. However, our results also indicate that combining
 1865 CNNs and GPs does not generally improve the performance of standard CNNs.
 1866 This can serve as a motivation for investigating new approximation methods
 1867 for scalable inference in GP models and combinations with CNNs.

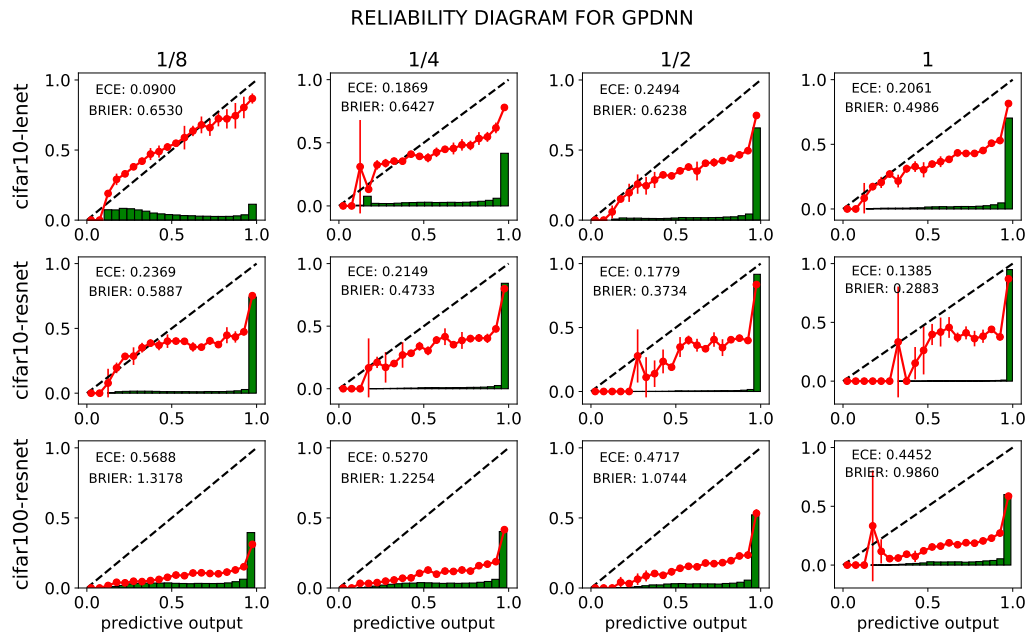


Figure 3.8 – Reliability diagrams for GPDNN

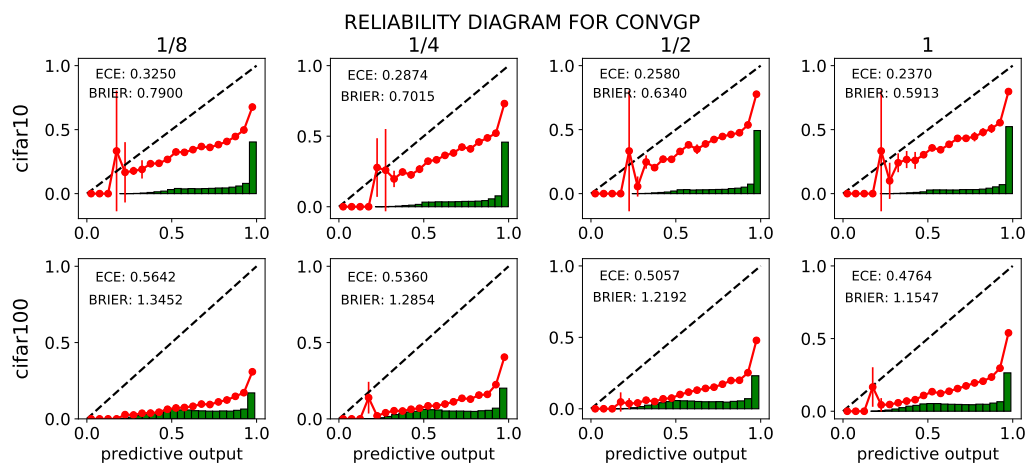


Figure 3.9 – Reliability diagrams for CGP

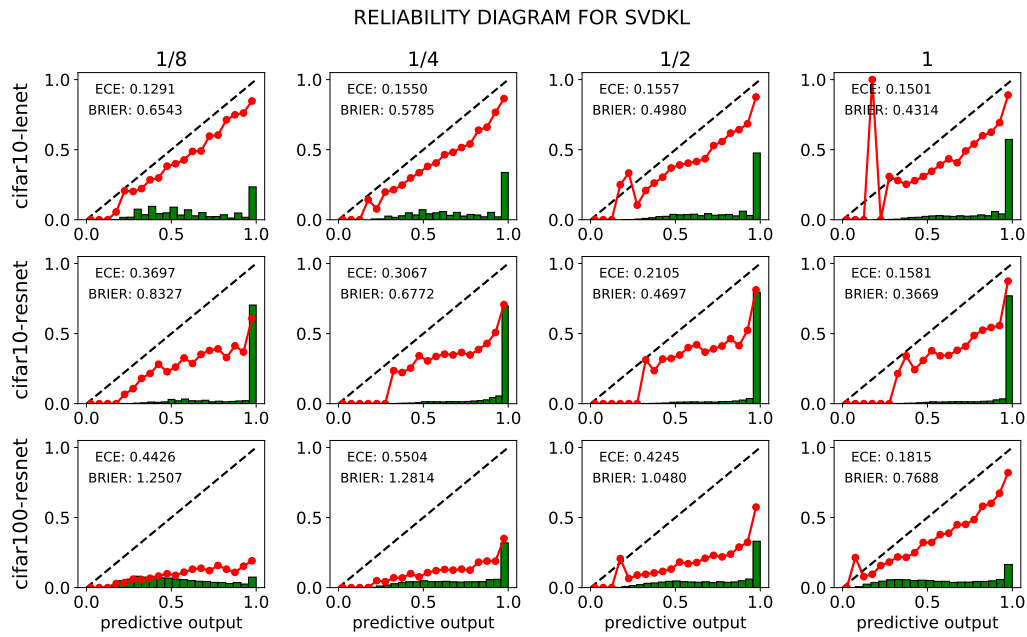


Figure 3.10 – Reliability diagrams for SVDKL

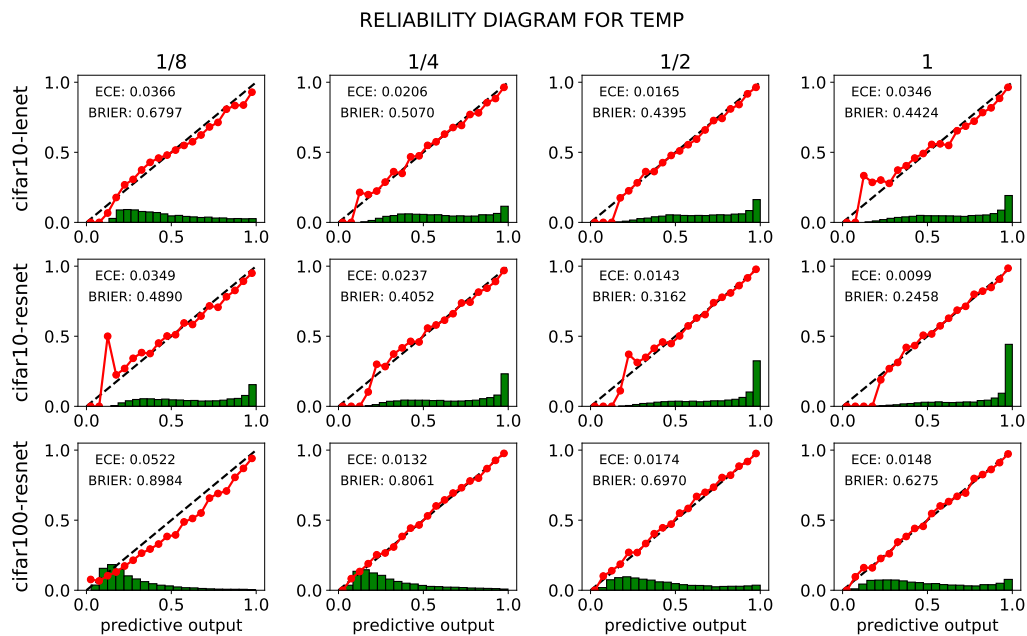


Figure 3.11 – Reliability diagrams for CNN+CAL

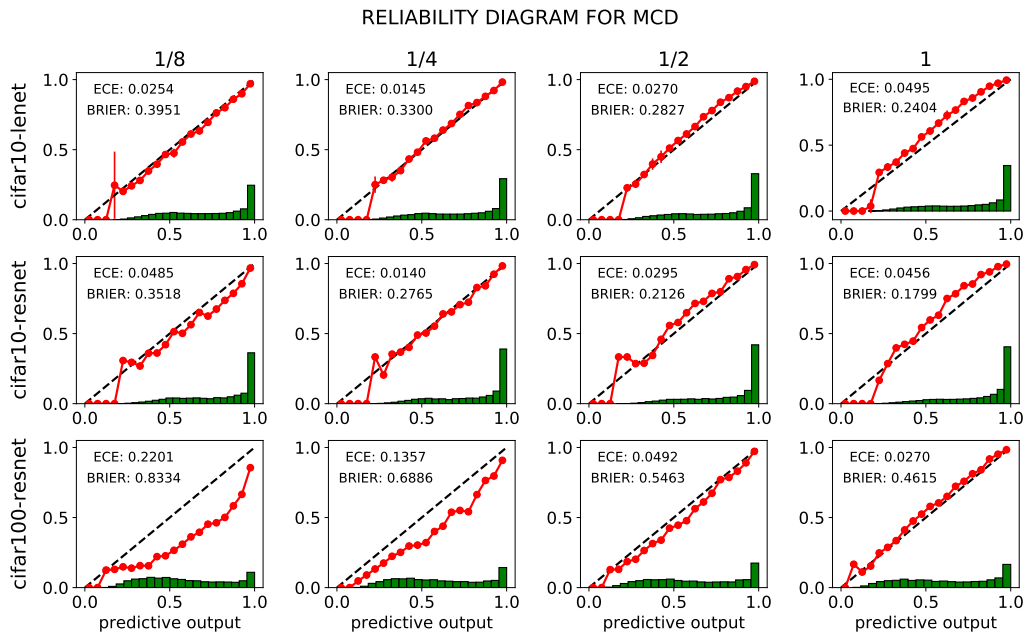


Figure 3.12 – Reliability diagrams for MCD

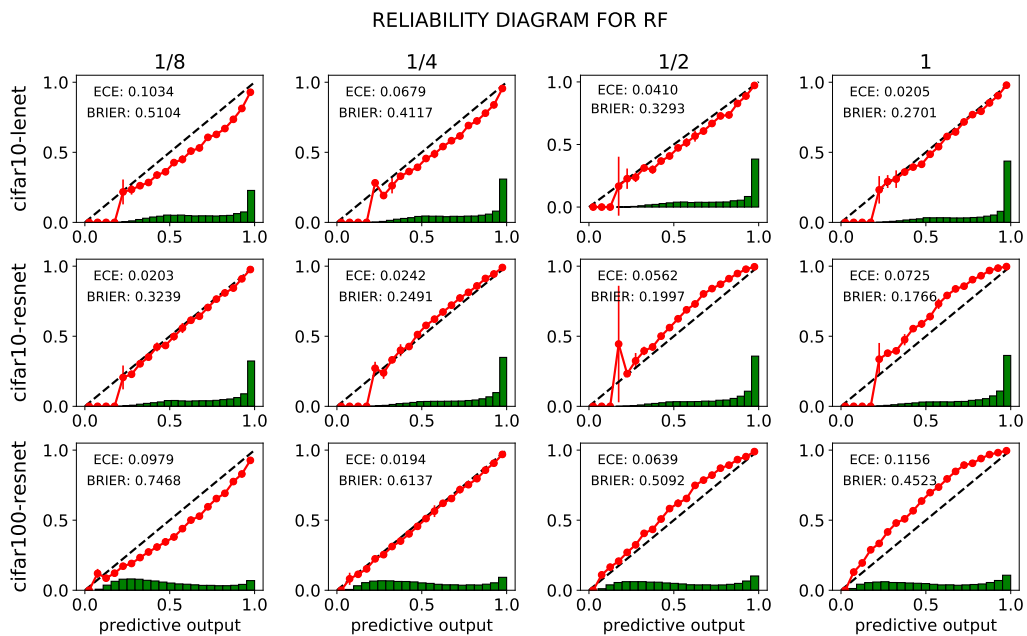


Figure 3.13 – Reliability diagrams for CNN+GP(RF)

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

1897 et al., 2013; Wilson and Nickisch, 2015; Hensman et al., 2015a). Some of
1898 the more recent approaches, such as Scalable Variational Gaussian Processes
1899 (SVGPs) (Hensman et al., 2015a), allow for the application of GPs to problems
1900 with millions of data points. In most applications of scalable GPs, these are
1901 approximated using M inducing points (IPs), which results in a complexity of
1902 $\mathcal{O}(M^3)$. It has been shown recently by Burt et al. (2019) that it is possible to
1903 obtain an arbitrarily good approximation for a certain class of GP models (i.e.
1904 conjugate likelihoods, concentrated distribution for the training data) with M
1905 growing more slowly than N . However, the general case remains elusive and
1906 it is still possible that the required value for M may exceed a certain compu-
1907 tational budget. Our result contributes to strengthen our belief that sparsity
1908 does not only enjoy desirable theoretical properties, but it also constitutes an
1909 extremely computationally efficient method in practice.

1910

1911 In this work, we push the limits of scalability and effectiveness of sparse GPs
1912 enabling a further reduction in complexity, which can be translated to higher
1913 accuracy by considering a larger set of inducing variables. The idea is to op-
1914 erate on a subset of H inducing points during training and prediction, with
1915 $H \ll M$, while maintaining a sparse approximation with M inducing vari-
1916 ables. We formalize our strategy by imposing a sparsity-inducing structure
1917 on the prior over the inducing variables and by carrying out a variational
1918 formulation of this model. This extends the original SVGP framework and
1919 enables mini-batch-based optimization for the variational objective. We then
1920 consider ways to select the set of H inducing points based on neighbor in-
1921 formation; at training time, for a given mini-batch, we activate H out of M
1922 inducing variables considering the nearest inducing inputs to the samples in
1923 the mini-batch, whereas at test time we select inducing variables correspond-
1924 ing to the inducing inputs which are nearest to the test data-points. We name
1925 our proposal *Sparse within a Sparse* GP (SWSGP). SWSGP is characterized by
1926 a number of attractive features: (i) it improves significantly the prediction
1927 quality using a small number of neighboring inducing inputs, and (ii) it ac-
1928 celerates the training phase, especially when the total number of inducing
1929 points becomes large. We extensively validate these properties on a variety
1930 of regression and classification tasks. We also showcase SWSGP on a large
1931 scale classification problem where we set $M = 100,000$; we are not aware of
1932 other approaches that can handle such a large set of inducing inputs without
1933 imposing some special structure on them (e.g., grid) or without considering
1934 one-dimensional inputs.

1935

1936 Hierarchical priors are often applied in Bayesian modeling to achieve com-
1937 pression and to improve flexibility (Molchanov et al., 2017; Louizos et al.,

1938 2017). To the best of our knowledge, this work is the first to explore these
1939 ideas for the purposes of sparsifying the inducing set in sparse GPs.

1940 4.2 Related work and background

1941 Sparse GPs that operate on inducing inputs have been extensively studied in
1942 the last 20 years (Csató and Opper, 2002; Lawrence et al., 2002; Snelson and
1943 Ghahramani, 2005; Quiñonero Candela and Rasmussen, 2005; Naish-Guzman
1944 and Holden, 2007). Many attempts on sparse GPs specified inducing inputs
1945 by satisfying certain criteria that produce an informative set of inducing vari-
1946 ables (Csató and Opper, 2002; Lawrence et al., 2002; Seeger et al., 2003).
1947 A different treatment has been proposed by Titsias (2009), which involves
1948 formulating the selection of inducing inputs as optimization of a variational
1949 lower bound to the marginal likelihood. The variational framework was later
1950 expanded so that stochastic optimization can be admitted, thus improving
1951 scalability for regression (Hensman et al., 2013) and classification (Hensman
1952 et al., 2015a). In a more recent work (Panos et al., 2018) scalability is ad-
1953 dressed in terms of the dimensionality of the input. All the aforementioned
1954 methodologies share a computational complexity of $\mathcal{O}(M^3)$. Although there
1955 have been some attempts in the literature to infer the appropriate number of
1956 inducing points as well as the inducing inputs (Pourhabib et al., 2014a; Burt
1957 et al., 2019), a large number of inducing variables is desirable in improving
1958 the approximation to the posterior. In this work we present a methodology
1959 that builds on the SVGP framework (Hensman et al., 2015a) and reduces its
1960 complexity, thus increasing the potential of sparse GP application on even
1961 larger datasets and with a larger set of inducing variables.

1962
1963 A different approach to scalable GPs was introduced by Wilson and Nickisch
1964 (2015), namely Kernel Interpolation for Scalable Structured GPs (KISS-GP).
1965 This line of work involves arranging a large number of inducing inputs into
1966 a grid structure; this allows one to scale to very large datasets by means of
1967 fast linear algebra. The applicability of KISS-GP on higher-dimensional prob-
1968 lems has been addressed by Wilson et al. (2015) by means of low-dimensional
1969 projections. A more recent extension allows for a constant-time variance pre-
1970 diction using Lanczos methods (Pleiss et al., 2018). Our work takes a different
1971 approach by keeping the GP prior intact, and by imposing sparsity on the set
1972 of inducing variables.

1973
1974 Local approximation of GPs inspired by the the concept of divide-and-conquer
1975 is also a practical solution to implement scalable GPs (Kim et al., 2005; Urta-

1976 sun and Darrell, 2008; Datta et al., 2016; ?; ?) which allows GPs to work on
 1977 large-scale datasets. In our work, we use neighbour information in a different
 1978 way, by incorporating it in a certain hierarchical structure of the auxiliary
 1979 variables through a variational scheme.

1980 4.2.1 Scalable Variational Gaussian Processes

1981 Consider a supervised learning problem with inputs $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^\top$ as-
 1982 sociated with labels $\mathbf{y} = (y_1, \dots, y_N)^\top$. Given a set of latent variables $\mathbf{f} =$
 1983 $(f_1, \dots, f_N)^\top$, GP models assume that labels are stochastic realizations based
 1984 on \mathbf{f} and a likelihood function $p(\mathbf{y} | \mathbf{f})$. In SVGPs, the set of inducing points
 1985 is characterized by inducing inputs $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_M)^\top$ and inducing variables
 1986 $\mathbf{u} = (u_1, \dots, u_M)^\top$. Regarding \mathbf{f} and \mathbf{u} , we have the following joint prior:

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N} \left(0, \begin{bmatrix} \mathbf{K}_{\mathbf{X}} & \mathbf{K}_{\mathbf{X}, \mathbf{Z}} \\ \mathbf{K}_{\mathbf{Z}, \mathbf{X}} & \mathbf{K}_{\mathbf{Z}} \end{bmatrix} \right), \quad (4.1)$$

1987 where $\mathbf{K}_{\mathbf{X}}$, $\mathbf{K}_{\mathbf{Z}}$ and $\mathbf{K}_{\mathbf{X}, \mathbf{Z}}$ are covariance matrices evaluated at the inputs
 1988 indicated by the subscripts. The posterior over inducing variables is approx-
 1989 imated by a variational distribution $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} | \mathbf{m}, \mathbf{S})$, while keeping the
 1990 exact conditional $p(\mathbf{f} | \mathbf{u})$ intact, that is $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} | \mathbf{u})q(\mathbf{u})$. The variational
 1991 parameters \mathbf{m} and \mathbf{S} , as well as the inputs \mathbf{Z} , are optimized by maximizing a
 1992 lower bound on the marginal likelihood $p(\mathbf{y} | \mathbf{X}) = \int p(\mathbf{y} | \mathbf{f})p(\mathbf{f} | \mathbf{X})d\mathbf{f}$. The
 1993 lower bound on $\log p(\mathbf{y} | \mathbf{X})$ can be obtained by considering the form of $q(\mathbf{f}, \mathbf{u})$
 1994 above and by applying Jensen’s inequality:

$$\mathbb{E}_{q(\mathbf{f})} \log p(\mathbf{y} | \mathbf{f}) - \text{KL}(q(\mathbf{u}) \| p(\mathbf{u})). \quad (4.2)$$

1995 The approximate posterior $q(\mathbf{f})$ can be computed by integrating out \mathbf{u} : $q(\mathbf{f}) =$
 1996 $\int q(\mathbf{u})p(\mathbf{f} | \mathbf{u})d\mathbf{u}$. Thanks to the Gaussian form of $q(\mathbf{u})$, $q(\mathbf{f})$ can be computed
 1997 analytically:

$$q(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{A}\mathbf{m}, \mathbf{K}_{\mathbf{X}} + \mathbf{A}(\mathbf{S} - \mathbf{K}_{\mathbf{Z}})\mathbf{A}), \quad (4.3)$$

1998 where $\mathbf{A} = \mathbf{K}_{\mathbf{X}, \mathbf{Z}}\mathbf{K}_{\mathbf{Z}}^{-1}$. When the likelihood factorizes over training points,
 1999 the lower bound can be re-written as:

$$\sum_{i=1}^N \mathbb{E}_{q(f_i)} [\log p(y_i | f_i)] - \text{KL}(q(\mathbf{u}) \| p(\mathbf{u})). \quad (4.4)$$

2000 Each term of the one-dimensional expectation of the log-likelihood can be com-
 2001 puted by Gauss-Hermite quadrature for any likelihoods (and analytically for
 2002 the Gaussian likelihood). The $\text{KL}(q(\mathbf{u}) \| p(\mathbf{u}))$ term can be computed ana-
 2003 lytically given that $q(\mathbf{u})$ and $p(\mathbf{u})$ are both Gaussian. To maintain positive-
 2004 definiteness of \mathbf{S} and perform unconstrained optimization, \mathbf{S} is parametrized
 2005 as $\mathbf{S} = \mathbf{L}\mathbf{L}^T$, with \mathbf{L} lower triangular.

2006 4.3 Sparse within Sparse Gaussian Processes

2007 We present a novel formulation of sparse GPs, which permits the use of a
 2008 random subset of the inducing points with little loss in performance. We
 2009 introduce a set of binary random variables $\mathbf{w} \in \{0, 1\}^M$ to govern the inclusion
 2010 of inducing inputs \mathbf{Z} and the corresponding variables \mathbf{u} . We then employ these
 2011 random variables to define a hierarchical structure on the prior as follows:

$$p(\mathbf{u} | \mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{D}_{\mathbf{w}} \mathbf{K}_{\mathbf{Z}} \mathbf{D}_{\mathbf{w}}), \quad (4.5)$$

2012 where $\mathbf{D}_{\mathbf{w}} = \text{Diag}(\mathbf{w})$, and $\mathbf{w} \sim p(\mathbf{w})$. Although the marginalized prior $p(\mathbf{u})$
 2013 is not Gaussian, it is possible to use the joint $p(\mathbf{u}, \mathbf{w}) = p(\mathbf{u} | \mathbf{w}) p(\mathbf{w})$ within a
 2014 variational scheme. We thus consider a random subset of the inducing points
 2015 during the evaluation of the prior in the variational scheme that follows; no
 2016 inducing points are permanently removed. Regarding $p(\mathbf{w})$, we consider an
 2017 implicit distribution: its analytical form is unknown, but we can draw samples
 2018 from it. Later, we will consider $p(\mathbf{w})$ based on the nearest inducing inputs to
 2019 random mini-batches of data.

2020 4.3.0.0.1 Remarks on the prior over \mathbf{f}

2021 Our strategy simply assumes a certain structure on the auxiliary variables,
 2022 but it *has no effect* on the prior over \mathbf{f} ; the latter remains unchanged. Let
 2023 \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
 2024 appropriate ordering, the conditional $\mathbf{u} | \mathbf{w}$ is effectively the element-wise
 2025 product $[\mathbf{u}_{\mathcal{I}}, \mathbf{u}_{\mathcal{J}}]^{\top} = \mathbf{u} \circ \mathbf{w}$. This reduces the variances and covariances of
 2026 some elements of \mathbf{u} to zero yielding a distribution of this form:

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

2027 The rows and columns of $\mathbf{u}_{\mathcal{J}}$ can simply be ignored. Regardless of the value
 2028 of \mathbf{w} , the conditional $\mathbf{f}, \mathbf{u}_{\mathcal{I}} | \mathbf{w}$ is always a Gaussian marginal, as it is a subset
 2029 of Gaussian variables. The marginalized $p(\mathbf{f}, \mathbf{u}) = \int p(\mathbf{f}, \mathbf{u} | \mathbf{w}) p(\mathbf{w}) d\mathbf{w}$ is
 2030 mixture of Gaussian densities, where the marginal over \mathbf{f} is the same for every
 2031 component of the mixture.

2032 The effect on \mathbf{f} is demonstrated in Figure 4.1, where we sample from the
 2033 (non-Gaussian) marginalized prior $p(\mathbf{u})$ in two steps: first we consider an
 2034 arbitrary random subset $\mathbf{u}_{\mathcal{I}}$, and then we sample from $p(\mathbf{u}_{\mathcal{I}}) \equiv p(\mathbf{u} | \mathbf{w})$.
 2035 Finally, \mathbf{f} samples are drawn from $p(\mathbf{f} | \mathbf{u}_{\mathcal{I}})$, which only involves the selected
 2036 inducing variables $\mathbf{u}_{\mathcal{I}}$. Following Eq. (4.1), the conditional $\mathbf{f} | \mathbf{u}$ is normally-
 2037 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 $\mathbf{S}_{\mathbf{f} | \mathbf{u}_{\mathcal{I}}} = \mathbf{K}_{\mathbf{X}} -$

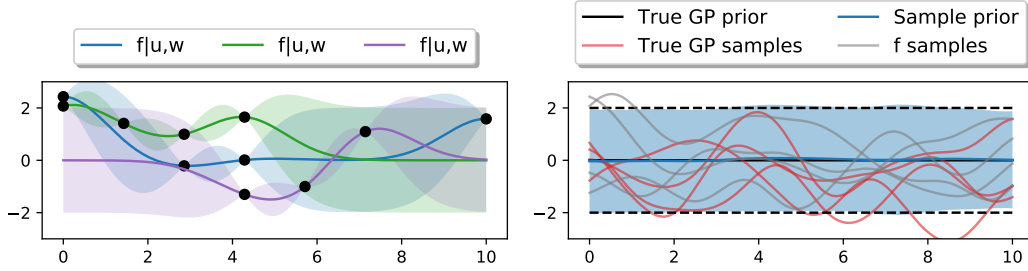


Figure 4.1 – The choice of inducing points does not affect the prior samples drawn from $p(\mathbf{f})$. Left: visualizations of $\mathbf{f} | \mathbf{u}, \mathbf{w}$ for different samples of \mathbf{w} . Right: comparison of the marginalised (w.r.t. \mathbf{u}, \mathbf{w}) prior over \mathbf{f} , against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

2038 $\mathbf{K}_{\mathbf{x}, \mathbf{z}_T} \mathbf{K}_{\mathbf{z}_T}^{-1} \mathbf{K}_{\mathbf{z}_T, \mathbf{x}}$. These conditionals can be seen for different samples of
 2039 \mathbf{u}, \mathbf{w} in the left side of Figure 4.1, while in the right side we compare the
 2040 marginalized prior over \mathbf{f} against the true GP prior.

2041 Of course, although the prior remains unchanged, that is not the case for the
 2042 posterior approximation. It is well known that the choice of inducing inputs
 2043 has an effect on the variational posterior (Titsias, 2009; Burt et al., 2019). Our
 2044 choice to impose a hierarchical structure to the inducing variables through \mathbf{w}
 2045 effectively changes the model compared to SVGP, and we adapt the variational
 2046 scheme accordingly.

2047 4.3.1 Lower bound on marginal likelihood

2048 By introducing \mathbf{u}, \mathbf{w} and using Jensen’s inequality, the lower bound on $\log p(\mathbf{y})$
 2049 can be obtained as follows

$$E_{q(\mathbf{u}, \mathbf{w})} \log p(\mathbf{y} | \mathbf{u}, \mathbf{w}) - \text{KL}(q(\mathbf{u}, \mathbf{w}) || p(\mathbf{u}, \mathbf{w})), \quad (4.7)$$

2050 where we choose the variational distribution q to reflect the hierarchical struc-
 2051 ture of the prior, i.e. $q(\mathbf{u}, \mathbf{w}) = q(\mathbf{u} | \mathbf{w}) p(\mathbf{w})$. This choice enforces sparsity
 2052 over the approximate posterior q ; the variational parameters are shared among
 2053 the conditionals $q(\mathbf{u} | \mathbf{w})$, for which we assume:

$$q(\mathbf{u} | \mathbf{w}) = \mathcal{N}(\mathbf{u} | \mathbf{D}_w \mathbf{m}, \mathbf{D}_w \mathbf{S} \mathbf{D}_w) \quad (4.8)$$

2054 By maximizing the variational bounds that follow, we impose a q that per-
 2055 forms well under a sparsified inducing set. We continue by applying Jensen’s
 2056 inequality on $p(\mathbf{y} | \mathbf{u}, \mathbf{w})$, obtaining:

$$\log p(\mathbf{y} | \mathbf{u}, \mathbf{w}) \geq E_{p(\mathbf{f} | \mathbf{u}, \mathbf{w})} \log p(\mathbf{y} | \mathbf{f}) \quad (4.9)$$

2057 We can now substitute (4.9) into (4.7), obtaining a bound where we expand
 2058 $q(\mathbf{u}, \mathbf{w})$ as $q(\mathbf{u} | \mathbf{w}) p(\mathbf{w})$. By making this assumption, we obtain the following
 2059 evidence lower bound $\mathcal{L}_{\text{ELBO}}$:

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

2060 Recall that $p(\mathbf{w})$ is implicit: although we do not make any particular as-
 2061 sumptions about its analytical form, we can draw samples from it. Using MC
 2062 sampling from $p(\mathbf{w})$, we can obtain the approximation $\tilde{\mathcal{L}}_{\text{ELBO}}$:

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

2063 where $\tilde{\mathbf{w}}^{(n)}$ is sampled from $p(\mathbf{w})$.

2064 4.3.1.0.1 Sampling from the set of inducing points.

2065 Recall that any sample $\tilde{\mathbf{w}}$ from $p(\mathbf{w})$ is a binary vector, i.e. $\mathbf{w} \in \{0, 1\}^M$. In
 2066 case all elements of \mathbf{w} are set to one, our approach recovers the original SVGP
 2067 with computational cost of $\mathcal{O}(M^3)$ coming from computing $p(f_n | \mathbf{u}, \tilde{\mathbf{w}} = \mathbf{1})$
 2068 and $\text{KL}(q(\mathbf{u} | \mathbf{w}) \parallel p(\mathbf{u} | \mathbf{w}))$ in the ELBO. When a \tilde{w}_i is set to zero, the entries
 2069 of the i -th row and i -th column of the covariance matrix in $p(\mathbf{u} | \mathbf{w})$ and
 2070 $q(\mathbf{u} | \mathbf{w})$ are zero. This means that the i -th variable becomes unnecessary, so
 2071 we get rid of i -th row and column in these matrices, and also eliminate the
 2072 i -th element in mean vectors of $q(\mathbf{u} | \mathbf{w})$ and $p(\mathbf{u} | \mathbf{w})$. This is equivalent to
 2073 selecting a set of active inducing points in each training iteration.

2074 4.3.2 H-nearest inducing inputs

2075 Despite the fact that $p(\mathbf{w})$ is an implicit distribution, we have been able to
 2076 define and calculate a variational bound, assuming we can sample from $p(\mathbf{w})$.
 2077 We shall now describe our sampling strategy, which relies on neighbor infor-
 2078 mation of random mini-batches.

2079
 2080 In order to explain the idea conveniently, we introduce $\mathbf{Z}_{\mathbf{x}}^H$ as the set of H -
 2081 nearest inducing inputs. Intuitively, the prediction for an unseen data \mathbf{x} using
 2082 $\mathbf{Z}_{\mathbf{x}}^H$ is a good approximation of the prediction using all M inducing points,

2083 that is $\mathbf{Z}_{\mathbf{x}}^M$. This can be verified by looking at the predictive mean, which is
 2084 expressed as a linear combination of kernel functions evaluated between train-
 2085 ing points and a test point, as in Eq. (4.3). The majority of the contribution
 2086 is given by the inducing points with the largest kernel values, so we can use
 2087 this as a criterion to establish whether an inducing input is “close” to an input
 2088 vector (the effect of different kernels on the definition of nearest neighbors
 2089 is explored in the supplement). With this intuition, $p(\mathbf{w})$ becomes a deter-
 2090 ministic function $w(\mathbf{x})$ indicating which inducing inputs are activated. For
 2091 mini-batch-based training, the value of \mathbf{w} remains random, as it depends on
 2092 the elements \mathbf{x} that are selected in the random mini-batch; this materializes
 2093 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 $\boldsymbol{\theta}$.

- 1: Initialize $\boldsymbol{\theta}$, i.e. kernel’s parameters, \mathbf{Z} , \mathbf{m} and \mathbf{S} .
 - 2: **while** stopping criteria is False **do**
 - 3: ELL \leftarrow 0 and KL \leftarrow 0.
 - 4: Sample mini-batch \mathcal{I} of size n from \mathcal{D} .
 - 5: **for** $(\mathbf{x}_i, y_i) \in \mathcal{I}$ **do**
 - 6: Find $\mathbf{Z}_{\mathbf{x}_i}^H$, i.e. the H -nearest \mathbf{Z} to \mathbf{x}_i .
 - 7: Compute $w(\mathbf{x}_i)$ using $\mathbf{Z}_{\mathbf{x}_i}^H$ as in (4.12)
 - 8: Extract $\mathbf{m}_{w(\mathbf{x}_i)}$ and $\mathbf{S}_{w(\mathbf{x}_i)}$ from \mathbf{m} and \mathbf{S} .
 - 9: Compute $q(f_i | w(\mathbf{x}_i))$ as in (4.13).
 - 10: ELL \leftarrow ELL + $\mathbb{E}_{q(f_i | w(\mathbf{x}_i))} \log p(y_i | f_i)$.
 - 11: KL \leftarrow KL + KL($q(\mathbf{u}_{w(\mathbf{x}_i)}) \| p(\mathbf{u}_{w(\mathbf{x}_i)})$)
 - 12: **end for**
 - 13: $\tilde{\mathcal{L}}_{\text{ELBO}} \leftarrow \frac{N}{n} \text{ELL} - \frac{1}{n} \text{KL}$.
 - 14: Update $\boldsymbol{\theta}$ using the derivative of $\tilde{\mathcal{L}}_{\text{ELBO}}$.
 - 15: **end while**
-

2094 test time, however, the inputs of interest are not random; we need to describe
 2095 the predictive distribution in terms of the deterministic function $w(\mathbf{x})$. In
 2096 fact, if we would like to approximate the predictive distribution at \mathbf{x}_n using
 2097 H -nearest inducing inputs to \mathbf{x} , i.e. $\mathbf{Z}_{\mathbf{x}_n}^H$, then $w(\mathbf{x}) = \left[w_{\mathbf{x}}^{(1)} \dots w_{\mathbf{x}}^{(M)} \right]^T$ where,

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

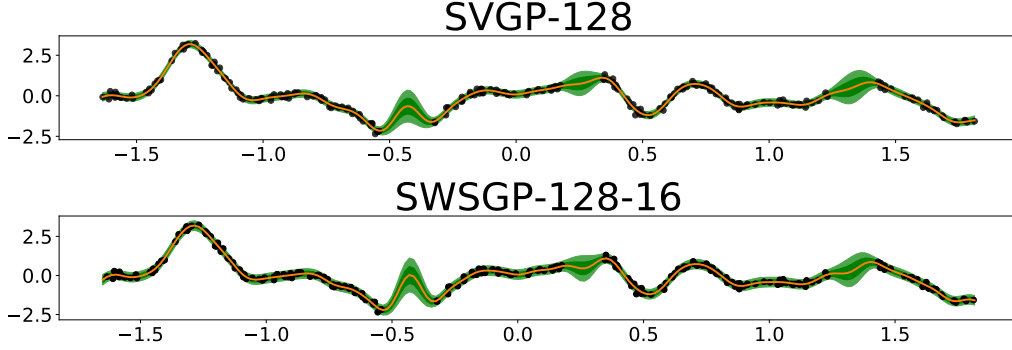


Figure 4.2 – Visualization of posterior distribution of SVGP and SWSGP. In both cases, we consider 128 inducing points; in terms of our scheme (SWSGP) we use 16 neighbors.

2099 We extract the relevant elements using $w(\mathbf{x})$; for the mean, we have $\mathbf{m}_{w(x_i)} =$
 2100 $\mathbf{D}_{w(x_i)} \mathbf{m}$, and for the covariance we select the appropriate rows and columns
 2101 using $\mathbf{S}_{w(x_i)} = \mathbf{D}_{w(x_i)} \mathbf{S} \mathbf{D}_{w(x_i)}$. The approximate posterior over f_i given $w(\mathbf{x}_i)$,
 2102 i.e. $q(f_i | w(\mathbf{x}_i))$ is:

$$\mathcal{N}\left(f_i \mid \mathbf{A}_{\mathbf{x}_i} \mathbf{m}_{w(x_i)}, \mathbf{K}_{\mathbf{x}_i} + \mathbf{A}_{\mathbf{x}_i} \left(\mathbf{S}_{w(x_i)} - \mathbf{K}_{\mathbf{Z}_{\mathbf{x}_i}^H} \right) \mathbf{A}_{\mathbf{x}_i}^\top\right), \quad (4.13)$$

2103 where $\mathbf{A}_{\mathbf{x}_i} = \mathbf{K}_{\mathbf{x}_i, \mathbf{Z}_{\mathbf{x}_i}^H} \mathbf{K}_{\mathbf{Z}_{\mathbf{x}_i}^H}^{-1}$.

2104
 2105 **One-dimensional regression example.** We visualize the posterior dis-
 2106 tribution for a synthetic dataset generated on a one-dimensional input space.
 2107 We execute SVGP and SWSGP, and depict the posterior distributions of these
 2108 two methods by showing the predictive means (orange lines) and the 95%
 2109 credible intervals (shaded areas) in Figure 4.2. We consider identical settings
 2110 for the two methods (i.e. 128 inducing points, kernel parameters, likelihood
 2111 variance) and a neighbor area of 16 for SWSGP; a full account of the setup
 2112 can be found in the supplement. We see that although the models are differ-
 2113 ent, the predictive distributions appear remarkably similar. A more extensive
 2114 evaluation follows in Section 4.4.

2115 4.3.3 Complexity

2116 The computational cost of SWSGP is dominated by lines 6, 8 and 9 in Algo-
 2117 rithm 2. For each data point (\mathbf{x}_i, y_i) in mini-batch \mathcal{I} , we need to find the
 2118 H nearest inducing neighbors $\mathbf{Z}_{\mathbf{x}_i}^H$ for n points in line 6, where $n = |\mathcal{I}|$; this
 2119 contributes to the worst-case complexity by $\mathcal{O}(nMH)$.

2120

2121 In line 8, we extract relevant parameters from \mathbf{m} and \mathbf{S} . We focus on the
 2122 cost of extracting $\mathbf{S}_{w(\mathbf{x}_i)}$ from \mathbf{S} . Similar to SVGP (Section 4.2.1), we con-
 2123 sider $\mathbf{S} = \mathbf{L}\mathbf{L}^T$, where \mathbf{L} is lower triangular. We extract $\mathbf{L}_{w(\mathbf{x}_i)} = \mathbf{D}_{w(\mathbf{x}_i)}\mathbf{L}$
 2124 which contains the rows of \mathbf{L} that correspond to the Cholesky decomposition
 2125 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 vari-
 2126 ational parameters is $\mathcal{O}(nMH^2)$.

2127

2128 Finally, the computation of approximating the predictive distribution in line
 2129 9 requires $\mathcal{O}(nH^3)$. The overall complexity for SWSGP in the general case
 2130 is $\mathcal{O}(nMH + nMH^2 + nH^3)$, which is a significant improvement over the
 2131 $\mathcal{O}(M^3)$ complexity of standard SVGP, assuming that $n, H \ll M$. If we choose
 2132 \mathbf{S} to be diagonal, the total complexity reduces to $\mathcal{O}(nMH + nH^3)$; if we ad-
 2133 ditionally consider \mathbf{Z} to be fixed, the computational cost is $\mathcal{O}(nH^3)$. In the
 2134 experiments of Section 4.4 we also explore these settings.

2135 4.4 Experiments

2136 In this section, we conduct experiments to evaluate SWSGP on a variety of
 2137 experimental conditions. We denote our approach by SWSGP-M-H, where M
 2138 inducing points are used and H determines how many neighbors are selected.
 2139 We introduce SVGP-M, SVGP-H and SVGP-M-H as competitors; SVGP-M and
 2140 SVGP-H are using M and H inducing points, respectively. SVGP-M-H, instead,
 2141 refers to SVGP using M inducing points at training time and H -nearest in-
 2142 ducing inputs at test time.

2143

2144 The comparison is carried out on some UCI data sets for regression and clas-
 2145 sification, i.e., POWERPLANT, KIN8NM, NAVAL, EEG, CREDIT, and SPAM. We
 2146 also consider larger scale data sets, such as MNIST and the AIRLINE data. We
 2147 use the Matérn- $5/2$ kernel in all cases except for the AIRLINE dataset, where
 2148 the sum of a Matérn- $3/2$ and a linear kernel is used, similar to Hensman et al.
 2149 (2015a). All models are trained using the Adam optimizer (Kingma and Ba,
 2150 2015) with a learning rate of 0.001 and a mini-batch size of 64. The likelihood
 2151 for regression and binary classification are set to Gaussian and probit func-
 2152 tion, respectively. All models are trained over 100,000 iterations except for
 2153 the AIRLINE data set where models are trained for one million iterations. In
 2154 regression tasks, we report the test root mean squared error (RMSE) and the
 2155 test mean negative log-likelihood (MNLL), whereas we report the test error
 2156 rate (ERR) and MNLL in classification tasks. The results are averaged over
 2157 three folds.

2158 4.4.1 Increasing the number of neighbors

2159 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

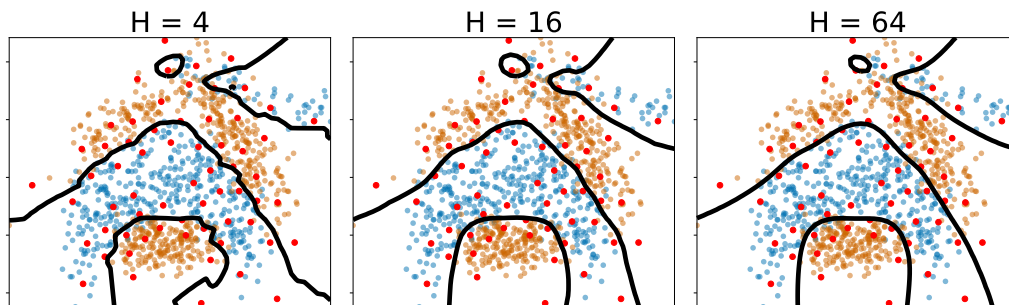


Figure 4.3 – Visualization of SWSGP on BANANA data sets with increasing H . The total number of inducing points M is fixed to 64, while the size of neighbor area H varies from 4 to 64. The red dots represent the inducing inputs. The orange and blue dots are training points from two different classes.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

2160 classification data set (BANANA), where M is fixed to 64 and H is increased
 2161 from 4 to 64. In general, these boundaries remain sensible across the whole
 2162 range of values of H , suggesting that SWSGP is able to work and converge well
 2163 even though H is significantly less than M . We also observe that the contours
 2164 of the classifier become smoother as H is increasing.

2165
 2166 We then test SWSGP on other data sets with larger dimensional inputs. In
 2167 these experiments, H is gradually increased to M . For POWERPLANT, KIN8NM,
 2168 NAVAL, EEG, CREDIT and SPAM, M is set to 64, and for MNIST and AIRLINE, M
 2169 is set to 512. In Fig. (4.4), we see that SWSGP-M-H consistently outperforms
 2170 SVGP-M-H and SVGP-H. This suggests that including neighbor information
 2171 at prediction time, combined with the use of a larger set of inducing points
 2172 alone is not enough to obtain competitive performance, and that only thanks
 2173 to the sparsity-inducing prior over latent variables, this yields improvements.
 2174 Crucially, the performance obtained by SWSGP are comparable with those
 2175 obtained by SVGP-M, while at each iteration only a subset of H out of M
 2176 inducing points are updated, carrying a significant complexity reduction.
 2177

2178 4.4.2 Increasing the number of inducing points

2179 In this set of experiments, we show that the performance SWSGP improves
 2180 when increasing the total number of inducing points, while keeping the num-
 2181 ber of active inducing points H fixed. We first illustrate this on the BANANA

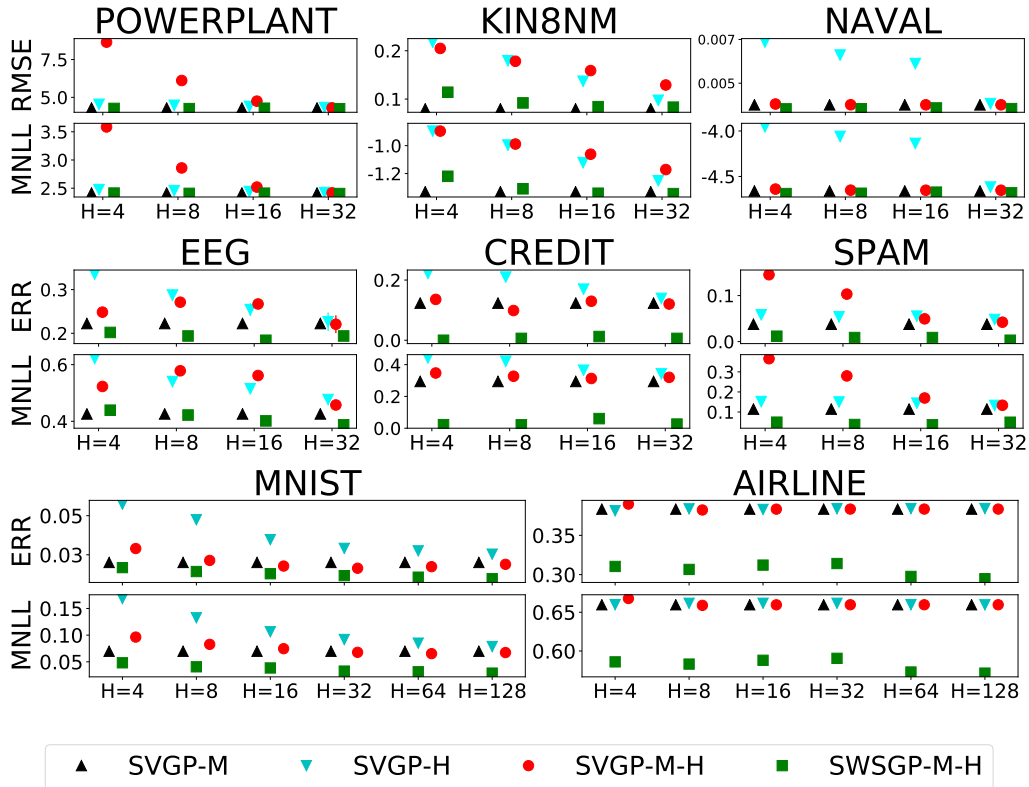


Figure 4.4 – Evaluation of SWSGP on high-dimensional data sets with increasing H . The black up-triangles are for SVGP with M inducing points, the cyan down-triangles are for SVGP with H inducing points, the red circles are for SVGP training with M inducing points and the prediction at an unseen data \mathbf{x} are made by \mathbf{Z}_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.

2182 data set, where H is fixed to 4 and M is gradually increased from 4 to 64. In
 2183 Fig. (4.5) we see that the classification boundaries improve when increasing
 2184 M .

2185

2186 We also investigate the impact of increasing H and M simultaneously. In
 2187 each regression and classification data set, we test SWSGP with $H = 4, 8$ and
 2188 $M = 8, 16, 32, 64$. The results shown in Fig. 4.6 indicate that using a small H
 2189 is not detrimental to performance when M is large. In addition, SWSGP with
 2190 a small H is comparable or better than SVGP in almost all cases.

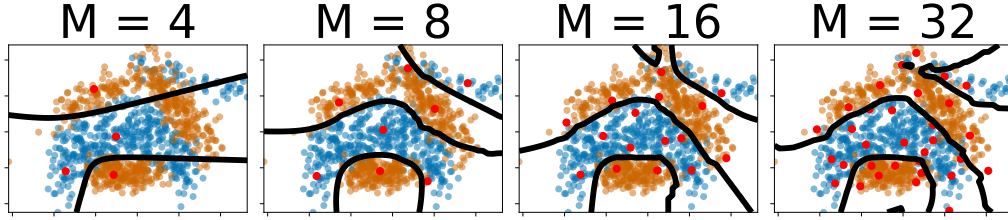
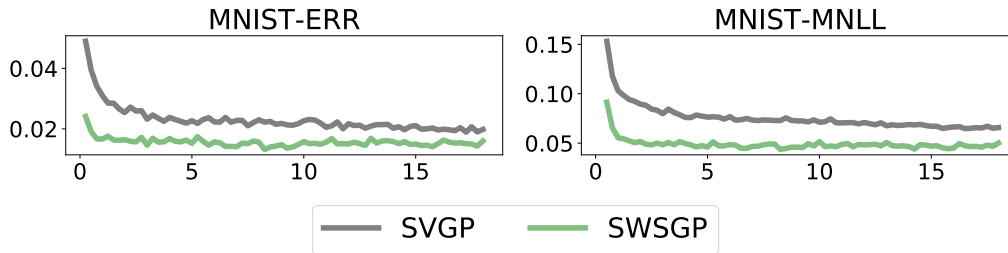


Figure 4.5 – Visualization of SWSGP on BANANA data sets with increasing M . The size of neighbor area H is set to 4. The total number of inducing points M varies from 4 to 64. The red dots represent inducing inputs. The orange and blue dots are the input points from the two different classes. The black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

2191 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



2192 We show the training and testing times of SWSGP and SVGP in Tab. 4.1. In
 2193 SVGP, we set $M = 256$ for POWERPLANT and KIN8NM, and 1024 for MNIST
 2194 and AIRLINE, i.e. SVGP-256 and SVGP-1024. In our approach, we use the same
 2195 M and we set H to 4 and M , i.e. SWSGP-256-4 and SWSGP-1024-4. Each cell
 2196 of Tab. 4.1 follows the format of $t_1 | t_2$ where t_1 and t_2 indicate execution time
 2197 of training and testing in milliseconds. The time t_1 is the averaged training
 2198 time of a training iteration. The time t_2 is the averaged execution time to
 2199 evaluate the predictive distribution on a test point. We stress that t_1 and t_2

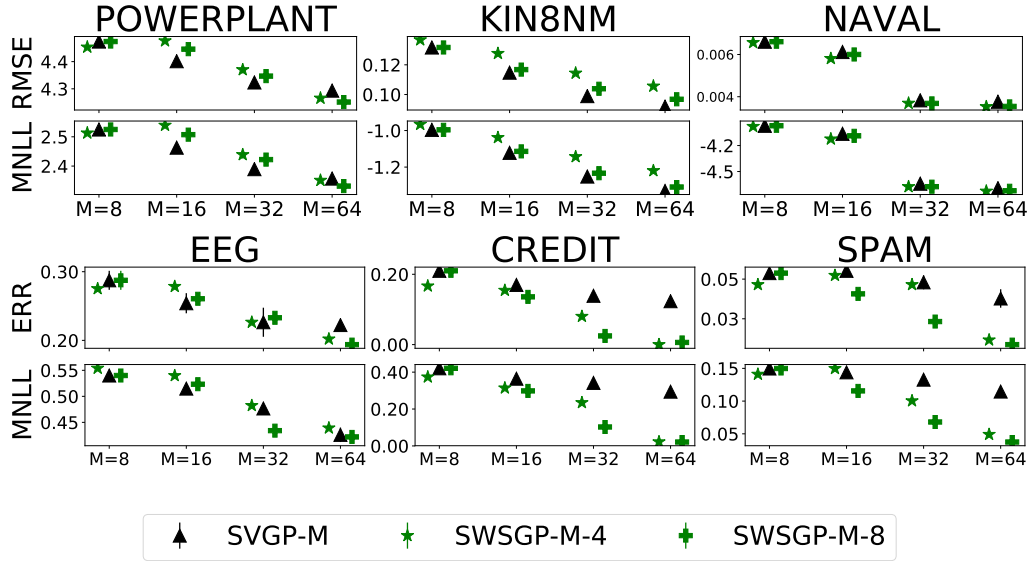


Figure 4.6 – Evaluation of SWSGP on high-dimensional data sets with increasing M . The black up-triangles are for SVGP with M inducing points. The green stars and plus are for SWSGP with H of 4 and 8 respectively. In these experiments, M varies from 4 to 64, as shown on horizontal axes. The standard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

2200 in SWSGP take into account the computation of finding neighbors inducing
 2201 inputs for each data point. In SVGP, we assume that $\mathbf{K}_{\mathbf{Z}}^{-1}$ is pre-computed
 2202 and saved after the training phase. Therefore, the computational cost to evalu-
 2203 ate the predictive distribution on a single test point is $\mathcal{O}(M^2)$. The time
 2204 t_2 in SVGP refers to the execution time of carrying out predictions with the
 2205 complexity of $\mathcal{O}(M^2)$.

2206

2207 The results in Tab 4.1 show a consistent improvement at test time compared
 2208 to SVGP across all values of H and M . At training time, the results show
 2209 a trend dependent on the number M of inducing points. Not surprisingly,
 2210 SWSGP offers limited improvements when M is small. Considering POWER-
 2211 PLANT and KIN8NM in which M is set to 256, SVGP is faster than SWSGP in
 2212 terms of training time. This is because the inversion of a 256×256 matrix
 2213 requires less time than finding the neighbors and inverting several 4×4 mat-
 2214 rices. However, Tab 4.1 shows dramatic speedups compared to SVGP when
 2215 the number of inducing points M is large. When $M = 1024$ on MNIST and
 2216 AIRLINE, SWSGP-1024-4 is faster than SVGP-1024 in training time. This is due
 2217 to the inversion of the 1024×1024 kernel matrix being a burden for SVGP,
 2218 whereas SWSGP deals with much cheaper computations. Finally, we show the

2219 progression of ERR and MNLL over training time when we train SVGP-1024
 2220 and SWSGP-1024-4 on MNIST. It becomes apparent that for large datasets our
 2221 method achieves high levels of accuracy significantly more quickly in terms of
 2222 running time compared to the standard SVGP.

2223 4.4.4 Large-scale problems with a huge number of IPs

2224 We showcase a large-scale classification problem, where we illustrate that
 2225 SWSGP enables the possibility to use sparse GPs with a massive number of
 2226 inducing points without incurring a prohibitive computational cost. We em-
 2227 ploy the AIRLINE data set, featuring 5 million training points. We test SWSGP
 2228 with $M = 100,000$ inducing points. We attempted to run SVGP with such a
 2229 large M without success (out of memory in a system with 32GB of RAM).
 2230 Therefore, as a baseline we report the results of SVGP with the configuration
 2231 in Hensman et al. (2015a).

2232
 2233 In SWSGP, we impose a diagonal matrix \mathbf{S} in the variational distribution
 2234 $q(\mathbf{u} | \mathbf{w})$, and we fix the position of the inducing inputs during training. By
 2235 fixing the inducing inputs, we can operate with pre-computed information
 2236 about which inducing inputs are neighbors of training inputs. Thanks to these
 2237 settings, SWSGP’s training phase requires $\mathcal{O}(nH^3)$ operations only, where n
 2238 is the mini-batch size. Due to the appropriate choice of H and n , and the
 2239 computational cost being independent of M , unlike SVGP, we can successfully
 2240 run SWSGP with $M = 100,000$.

2241
 2242 By setting H and the mini-batch size n to 100 and 16 respectively, in about 24
 2243 hours of training we could run SWSGP-100,000-100 for one million iterations.
 2244 The ERR and MNLL of SWSGP-100,000-100 evaluated on the test set are 21%
 2245 and 0.48, respectively, while the ERR and MNLL of SVGP-200 published in
 2246 Hensman et al. (2015a) are about 34% and 0.61, respectively. To the best of
 2247 our knowledge, SWSGP is the first to enable sparse GPs with such a large set
 2248 of inducing points without imposing a grid structure on the inducing inputs.
 2249 We conclude by reporting comparisons with other GP-based models. In par-
 2250 ticular, we compare against the Stochastic Variational Deep Kernel Learning
 2251 (SVDKL) (Wilson et al., 2016) and the Deep GP approximated with random
 2252 features (Cutajar et al., 2017). In the former, KISS-GP is trained on top of
 2253 a deep neural network which is optimized during training, and in the latter
 2254 the layers of a deep GP are approximated as parametric models using random
 2255 feature expansions. Both competitors feature mini-batch-based learning, so
 2256 this represents a challenging test for SWSGP. The results in Tab. 4.2 show
 2257 that SWSGP is comparable with these competitors. We believe that this is

2258 a remarkable result obtained by our shallow SWSGP, supporting the conclu-
 2259 sions of previous works showing that advances in kernel methods can result in
 2260 performance which are competitive with deep learning approaches (see, e.g.,
 2261 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)

2262 4.4.5 Comparison to Local GPs

2263 We finally demonstrate that SWSGP behaves differently from other approaches
 2264 that use local approximations of GPs. We consider two well-established ap-
 2265 proaches of local GPs proposed by Kim et al. (2005) and Urtasun and Darrell
 2266 (2008). Following Liu et al. (2018b), we shall refer to these methods as *In-*
 2267 *ductive* GPs and *Transductive* GPs, respectively. We run all methods on two
 2268 regression data sets: POWERPLANT and KIN8NM. We set the number of local
 2269 experts to 64, and we use the same number of inducing points for SWSGP
 2270 (with H either 4 or 8). As the size of POWERPLANT and KIN8NM are approx-
 2271 imately 7000, we set the number of training points governed by a local expert
 2272 to 100. For the local GP approaches, we choose 64 locations in the input space
 2273 using the K -means algorithm, and for each location we choose 100 neighbor-
 2274 ing points; we then train the corresponding local GP expert. Regarding the
 2275 testing phase, inductive GPs simply rely on the nearest local experts to an
 2276 unseen point \mathbf{x}_* . Whereas for transductive GPs, we use 100 neighbors of \mathbf{x}_*
 2277 and the nearest local expert to make predictions. In table 4.3, we summarize
 2278 RMSE and MNLL for all methods; SWSGP clearly outperforms the local GP
 2279 approaches in terms of MNLL.

Method	POWERPLANT	KIN8NM
	RMSE MNLL	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

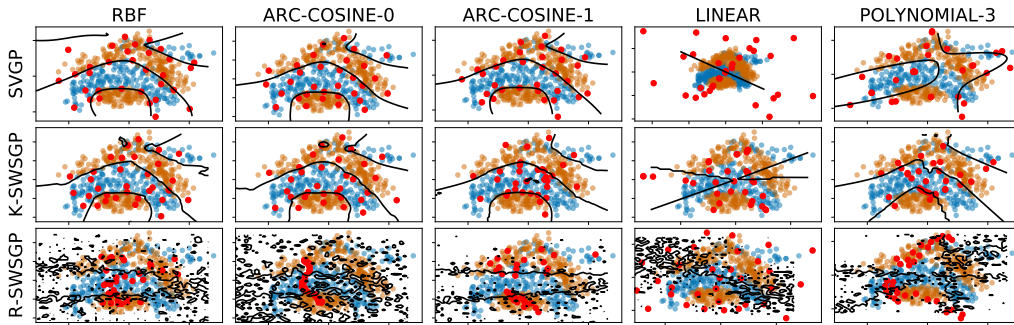


Figure 4.7 – SWSGP on various kernels and strategies for selecting the H-nearest inducing points.

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 neighborhood. Thus, the selected neighborhood 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 neighborhood-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

2297 Euclidean distance. We also examine a random-based heuristic (R-SWSGP) in
 2298 which H -nearest inducing points are randomly chosen. In all cases, we set M
 2299 and H as 32 and 8 respectively. We also compare against SVGP with M of 32.

2300

2301 In Fig. (4.7), we visualize the contours of classifiers of SVGP and SWSGP
 2302 with various configurations. Clearly, R-SWSGP does not work, i.e. the con-
 2303 tours are discontinuous and the locations of contours does not makes sense.
 2304 Regarding the kernels RBF, ARC-COSINE-0 and ARC-COSINE-1, our method
 2305 (K-SWSGP) seems to be virtually identical to SVGP. The advantages of K-
 2306 SWSGP over SVGP are shown when using POLYNOMIAL-3. It is highly possible
 2307 that the flexibility of variational distribution over inducing variables, i.e. $q(\mathbf{u})$,
 2308 in SWSGP is the main reason for this difference.

2309 4.5.2 Further visualizations on 1D examples

2310 We demonstrate SWSGP on one-dimensional regression problem. We have gen-
 2311 erated a synthetic data-set by sampling inputs x_i from the interval $[-2, 2]$; the
 2312 targets have been computed as $y_i = \sin(12x_i) + 0.66 \cos(25x_i) + \varepsilon$, where ε is
 2313 additive Gaussian noise with variance 0.1. Figure 4.8 summarizes the regres-
 2314 sion result for a fixed M , while the value of H varies from 4 to 64. We notice
 2315 that the predictive means are nearly identical across the different sub-figures.
 2316 These observations suggest that SWSGP is able to work and converge well even
 though H is significantly less than M .

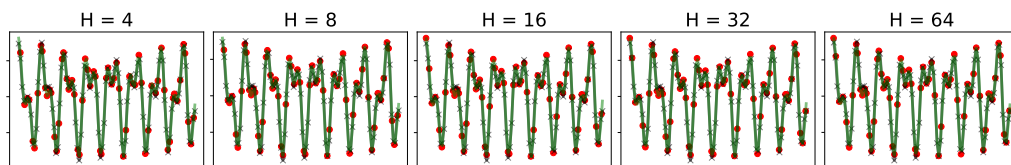


Figure 4.8 – SWSGP is applied on a one-dimensional data set, where M is fixed to 64 and H is increased gradually from 4 to 64. The red dots are inducing positions; the black crosses are testing points; the green line refers to predictive means.

2317

2318

2319 We also show that the performance of SWSGP improves when increasing the
 2320 total number of inducing points while keeping the number of active inducing
 2321 points H fixed. We intuitively expect that a larger the total number of induc-
 2322 ing points should translate to a more accurate model. In these experiments,
 2323 the size of neighbor area is fixed to 4, i.e. $H = 4$, and the total number
 2324 of inducing points are varies from 4 to 64. We see that the sequence of the
 2325 predictive means in Fig. 4.9 are more and more accurate from left to right.

2326 Although we are using a small neighbor area, our model is improved when
 2327 increasing the total number of inducing points.

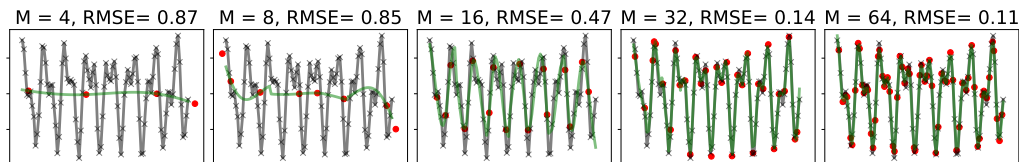


Figure 4.9 – SWSGP is applied on 1D. The red dots are inducing positions. The black crosses are testing samples. The green lines are predictive means. The title of each sub-figures shows M and corresponding RMSE.

2327

2328 4.6 Conclusions

2329 Sparse approaches that rely on inducing points have met with success in re-
 2330 ducing the complexity of GP regression and classification. However, these
 2331 methods are limited by the number of inducing inputs that is required to
 2332 obtain an accurate approximation of the true GP model. A large number of
 2333 inducing inputs is often necessary in cases of very large datasets, which marks
 2334 the limits of practical applications for most GP-based approaches.

2335

2336 In this work, we further improve the computational gains of sparse GPs.
 2337 We proposed SWSGP, a novel methodology that imposes a hierarchical and
 2338 sparsity-inducing effect on the prior over the inducing variables. This has
 2339 been realized as a conditional GP given a random subset of the inducing points,
 2340 which is defined as the nearest neighbors of random mini-batches of data. We
 2341 have developed an appropriate variational bound which can be estimated in
 2342 an unbiased way by means of mini-batches. We have performed an extensive
 2343 experimental campaign that demonstrated the superior scalability properties
 2344 of SWSGP compared to the state-of-the-art.

Conclusion

2345

2346

2347

2348 The models and techniques presented in this thesis are unified by the overar-
2349 ching goal of improving the calibration and scalability of Gaussian Processes.
2350 We conclude this thesis by summarizing the principal themes and contribu-
2351 tions presented in the preceding chapters, with particular emphasis on their
2352 significance in the context of complementary work in this direction of research.
2353 This is followed by a brief outlook on possible avenues for future work where
2354 we indicate how one might go about achieving these objectives.

2355 5.1 Themes and Contributions

2356 In this thesis, we primarily investigated the following themes in relation to
2357 Gaussian processes:

2358

2359 • **Well-calibrated deep convolutional probabilistic model.** Developing
2360 models which are able to provide accurate predictions and reliable uncertain-
2361 ties has been a long-standing research topic attracting significant attention
2362 from machine learning community. Deep CNNs that have accomplished state-
2363 of-the-art results in a range of tasks have been illustrated to be miscalibrated,
2364 the depth of architecture are the main factor affecting calibration (Guo et al.,
2365 2017). Thinking of Bayesian priors as a form of regularization, it is natural to
2366 assume that Bayesian CNNs are an appropriate treatment for the problem of
2367 miscalibration of modern CNNs. Independently of the works on Bayesian CNNs
2368 implemented by Monte Carlo Dropout (Gal and Ghahramani, 2016b), there
2369 have been other attempts to give a probabilistic flavor to CNNs by combining
2370 them with Gaussian processes (Wilson et al., 2016; Bradshaw et al., 2017;
2371 van der Wilk et al., 2017). To the best of our knowledge, prior to our work
2372 there were no studies showing calibration properties of these Bayesian CNNs
2373 approaches. Hence, in Chapter 3, we investigated the calibration properties of
2374 Bayesian treatment on CNNs. Perhaps surprisingly, the results indicated that
2375 current combinations of CNNs and GPs are miscalibrated, with a tendency of

2376 being overconfident in predictions. Consequently, by extending the random
2377 feature expansion approximation for DGPs (Cutajar et al., 2017), we proposed
2378 a novel combination of CNNs and GPs which is well-calibrated, and we val-
2379 idated it through several experimental results on image classification tasks.
2380 Furthermore, our model was extended by replacing the last fully-connected
2381 layers of CNNs with Deep GPs (Cutajar et al., 2017) and by employing struc-
2382 tured random features to obtain faster and more compact GP approximations
2383 (Le et al., 2013; Yu et al., 2016).

2384

2385 • **Combination of global and local approximation.** Gaussian Processes
2386 Rasmussen and Williams (2006) offer a powerful statistical framework for in-
2387 ference on functions. However, the applicability of GPs on big datasets is hin-
2388 dered by the prohibitive complexity depending on training size N . Due to the
2389 rigorous uncertainty quantification of GPs, the inducing point-based sparse
2390 approximation of GPs have been extensively studied (Snelson and Ghahra-
2391 mani, 2005; Quiñonero Candela and Rasmussen, 2005; Titsias, 2009; Hens-
2392 man et al., 2015a). The state-of-the-art approaches, e.g Scalable Variational
2393 Gaussian Processes (Hensman et al., 2015a), allows for the application of GPs
2394 to large-scale problems with a small number of inducing points M . As shown
2395 recently by Burt et al. (2019), it is possible to obtain an arbitrarily good
2396 approximation for a certain class of GP models with M growing more slowly
2397 than N . However, in general, it is still possible that the required value for
2398 M may exceed a certain computational budget. In Chapter 4, by imposing
2399 a sparsity-inducing structure on the prior over the inducing variables and by
2400 carrying out a variational formulation of this model, we pushed the limits
2401 of scalability and effectiveness of sparse GPs enabling a further reduction of
2402 computational complexity. Our experimental results showed that the use of
2403 unprecedented number of inducing points led to higher accuracy on AIRLINE
2404 which is a dataset with millions data points. In addition, we showed that our
2405 proposed model is able to know what it does not know by yielding sensible
2406 predictive uncertainties.

2407 5.2 Future work

2408 Beyond the discussion featured in this thesis, the themes explored in this body
2409 of work not only motivate immediate extensions for improvements, but also
2410 set the foundations for broader long-term objectives. In this section, we ex-
2411 pand upon the directions for future work which we believe to be particularly
2412 pertinent to ongoing developments in both the theoretical and practical as-
2413 pects of machine learning using GPs. We partition this discussion into the

2414 overarching themes of (i) studying calibration properties of GPs regression;
2415 and (ii) proposing more elegant mixtures of CNNs and GPs; and (iii) adapting
2416 the state-of-the-art scalable GPs to online machine learning.

2417 5.2.1 Calibrated GP regression

2418 In addition to improving the scalability of GPs, producing reliable predictive
2419 uncertainties is also a primary goal for the application of GPs in the era of big
2420 data, especially when GPs are components of larger decision-making pipelines.
2421 This aspect can be evaluated by analyzing calibration properties mentioned
2422 in Chapter 3. While the reliability of the predictive uncertainties of Bayesian
2423 CNNs on classification tasks has been analyzed Guo et al. (2017); Lakshmi-
2424 narayanan et al. (2017); Tran et al. (2019), the calibration of GP-based regres-
2425 sion methods has not been considered carefully. As mentioned in Kuleshov
2426 et al. (2018), the calibration property of regressors is evaluated by their pre-
2427 dictive interval. A regressor is stated to be calibrated if p -percent credible
2428 intervals contain the true outcomes p -percent of the time. Starting with the
2429 novel vision about reliable regressors, investigating calibration properties of
2430 GPs on regression promises to be interesting. Some potential candidates re-
2431 inforcing the model’s calibration may be inspired by the preceding works, for
2432 example post-calibration by Platt scaling (Platt, 1999; Guo et al., 2017) or
2433 training with adversarial samples (Lakshminarayanan et al., 2017).

2434 5.2.2 Elegant mixtures of CNNs and GPs

2435 While studying on Bayesian CNN, we have realized that combining CNNs and
2436 GPs does not generally improve the performance of standard GPs. We spec-
2437 ulate that the kernel’s parameterization with a high number of parameter
2438 increases the risk of overfitting, and leads to overconfident tendency in pre-
2439 dictions. As shown in Chapter 3, the Bayesian treatment on convolutional
2440 parameters enhances not only model’s generalization but also model’s cali-
2441 bration. However, the improvements of our approach carries a great compu-
2442 tational cost due to repeated feed-forward procedure. This limit can serve as
2443 a motivation for investigating new approximation methods for scalable infer-
2444 ence in GP models and combinations with CNNs.

2445
2446 Generally, the Bayesian flavor in the mixtures of CNNs and GPs can be strength-
2447 ened by applying a full Bayesian treatment. For example, following our works
2448 in Chapter 3, the proposed models can be further improved by applying a
2449 Bayesian treatment on priors of parameters, which would result in the opti-
2450 mization of dropout rates of convolutional hyperparameters (Kingma et al.,

2451 2015; Molchanov et al., 2017; Louizos et al., 2017). Along a similar vein, in-
2452 dependent works replacing the fully-connected layers of CNNs by GPs (Wilson
2453 et al., 2016; Bradshaw et al., 2017; Tran et al., 2019), while Deep Convolutional
2454 Gaussian Processes (DCGPs) proposed by Blomqvist et al. (2018) substitutes
2455 GPs for convolutional filters. Another interesting approach could be applying
2456 the Bayesian formulation mentioned in Titsias and Lazaro-Gredilla (2013) to-
2457 gether with sparsity inducing priors (Louizos et al., 2017; Molchanov et al.,
2458 2017) on DCGPs, a procedure of learning architecture is proposed, which not
2459 only accelerates computations but also allows one to approximately integrate
2460 out kernel hyperparameters, such as length-scales.

2461 5.2.3 Adaptability to online machine learning

2462 According to the extensive literature review in Liu et al. (2018b), local approx-
2463 imations are common approaches to implement scalable statistical inference
2464 systems. The uses of local approximations require to define the localization
2465 of experts, which directly affects to the assignments of data points to local
2466 experts. Likewise, in chapter 4, our proposal named Sparse-within-sparse
2467 Gaussian Processes (SWSGP) perceived as a combination of global and local
2468 approximations also relies on the way to select active inducing points for each
2469 inputs. On offline tasks, SWSGP was shown to be effective in terms of accu-
2470 racy and complexity. With the application of online machine learning wherein
2471 training sets are constantly evolving, the selection of active inducing points
2472 based upon spatial or temporal distance, which is implemented in SWSGP, may
2473 ignore the information related to periodic patterns. In such scenario, a kernel-
2474 based distance seems to be more appropriate because the kernel intuitively
2475 determines the correlation between two points in the input space. More gen-
2476 eral, by perceiving the selection of active inducing points as a gating function,
2477 the input-dependent Dirichlet Process (Rasmussen and Ghahramani, 2002)
2478 and Polya urn distribution (Meeds and Osindero, 2006) can automatically in-
2479 fer which inducing points are necessary from data. Another problem in the
2480 scenario of online machine learning is to define a scheme for removing unnec-
2481 essary inducing points. This can be done simply by eliminating the oldest
2482 ones. More elegantly, the frameworks proposed by McIntire et al. (2016); Bijl
2483 et al. (2016) could be employed.

Bibliography

- 2485 M. Abramowitz. *Handbook of Mathematical Functions, With Formulas,*
2486 *Graphs, and Mathematical Tables,*. Dover Publications, Inc., USA, 1974.
2487 ISBN 0486612724.
- 2488 D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané.
2489 Concrete problems in ai safety, 2016.
- 2490 Y. Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*,
2491 2(1):1–127, Jan. 2009. ISSN 1935-8237.
- 2492 J. Bernardo and A. Smith. *Bayesian Theory*, volume 15. 01 2000. ISBN 0
2493 471 49464 X.
- 2494 H. Bijl, T. B. Schön, J. Wingerden, and M. Verhaegen. Online sparse gaussian
2495 process training with input noise. *ArXiv*, abs/1601.08068, 2016.
- 2496 D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A
2497 review for statisticians. *Journal of the American Statistical Association*,
2498 112(518):859–877, 2017.
- 2499 K. Blomqvist, S. Kaski, and M. Heinonen. Deep convolutional gaussian pro-
2500 cesses, 2018.
- 2501 C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Un-
2502 certainty in Neural Network. In F. R. Bach and D. M. Blei, editors, *Pro-*
2503 *ceedings of the 32nd International Conference on Machine Learning, ICML*
2504 *2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and*
2505 *Conference Proceedings*, pages 1613–1622. JMLR.org, 2015.
- 2506 J. Bradshaw, Alexander, and Z. Ghahramani. Adversarial Examples, Uncer-
2507 tainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep
2508 Networks, July 2017. arXiv:1707.02476.
- 2509 G. W. Brier. VERIFICATION OF FORECASTS EXPRESSED IN TERMS
2510 OF PROBABILITY. *Monthly Weather Review*, 78(1):1–3, 01 1950. ISSN
2511 0027-0644.
- 2512 F.-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic.
2513 Probabilistic integration: A role in statistical computation?, 2015.

- 2514 D. Burt, C. E. Rasmussen, and M. Van Der Wilk. Rates of convergence for
2515 sparse variational Gaussian process regression. In *Proceedings of the 36th*
2516 *International Conference on Machine Learning*, volume 97 of *Proceedings*
2517 *of Machine Learning Research*, pages 862–871. PMLR, 2019.
- 2518 R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold
2519 Gaussian Processes for regression. In *2016 International Joint Conference*
2520 *on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29,*
2521 *2016*, pages 3338–3345, 2016.
- 2522 K. Chalupka, C. K. I. Williams, and I. Murray. A framework for evaluating
2523 approximation methods for Gaussian process regression. *Journal of Machine*
2524 *Learning Research*, 14, 2013.
- 2525 Z. Chen, J. Ma, and Y. Zhou. A precise hard-cut em algorithm for mixtures
2526 of gaussian processes. pages 68–75, 08 2014. ISBN 978-3-319-09338-3.
- 2527 Y. Cho and L. K. Saul. Kernel Methods for Deep Learning. In Y. Ben-
2528 gio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta,
2529 editors, *Advances in Neural Information Processing Systems 22*, pages 342–
2530 350. Curran Associates, Inc., 2009.
- 2531 C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20:
2532 273–297, 1995.
- 2533 T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans.*
2534 *Inf. Theor.*, 13(1):21–27, Sept. 2006. ISSN 0018-9448.
- 2535 L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computa-*
2536 *tion*, 14(3):641–668, 2002. ISSN 0899-7667.
- 2537 K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. Random fea-
2538 ture expansions for deep Gaussian processes. In *Proceedings of the 34th*
2539 *International Conference on Machine Learning*, volume 70 of *Proceedings*
2540 *of Machine Learning Research*, pages 884–893. PMLR, 2017.
- 2541 A. C. Damianou and N. D. Lawrence. Deep Gaussian Processes. In *Proceed-*
2542 *ings of the Sixteenth International Conference on Artificial Intelligence and*
2543 *Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*,
2544 volume 31 of *JMLR Proceedings*, pages 207–215. JMLR.org, 2013.
- 2545 A. Datta, S. Banerjee, A. Finley, and A. Gelfand. On nearest-neighbor gaus-
2546 sian process models for massive spatial data: Nearest-neighbor gaussian
2547 process models. *Wiley Interdisciplinary Reviews: Computational Statistics*,
2548 8, 08 2016.

- 2549 A. P. Dawid. The well-calibrated bayesian. 1982.
- 2550 M. H. DeGroot and S. E. Fienberg. The comparison and evaluation of fore-
2551 casters. *The Statistician: Journal of the Institute of Statisticians*, 32:12–22,
2552 1983.
- 2553 M. P. Deisenroth and J. W. Ng. Distributed gaussian processes. In *Pro-
2554 ceedings of the 32nd International Conference on International Conference
2555 on Machine Learning - Volume 37*, ICML’15, page 1481–1490. JMLR.org,
2556 2015.
- 2557 M. P. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-
2558 efficient approach to policy search. In *ICML*, 2011.
- 2559 P. Diggle and P. Ribeiro. *Model-based Geostatistics*. Springer Series in Statis-
2560 tics. Springer, mar 2007. ISBN 0387329072 978-0387329079.
- 2561 D. K. Duvenaud, O. Rippel, R. P. Adams, and Z. Ghahramani. Avoiding
2562 pathologies in very deep networks. In *Proceedings of the Seventeenth In-
2563 ternational Conference on Artificial Intelligence and Statistics, AISTATS
2564 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop
2565 and Conference Proceedings*, pages 202–210. JMLR.org, 2014.
- 2566 M. Filippone, M. Zhong, and M. Girolami. A comparative evaluation of
2567 stochastic-based inference methods for gaussian process models. *Machine
2568 Learning*, 93, 10 2013.
- 2569 P. A. Flach. Classifier Calibration. In C. Sammut and G. I. Webb, editors,
2570 *Encyclopedia of Machine Learning and Data Mining*, pages 1–8. Springer
2571 US, Boston, MA, 2016.
- 2572 S. Flaxman, A. G. Wilson, D. B. Neill, H. Nickisch, and A. J. Smola. Fast
2573 kronecker inference in gaussian processes with non-gaussian likelihoods. In
2574 *Proceedings of the 32nd International Conference on International Confer-
2575 ence on Machine Learning - Volume 37*, ICML’15, page 607–616. JMLR.org,
2576 2015.
- 2577 Y. Gal and Z. Ghahramani. Dropout As a Bayesian Approximation: Rep-
2578 resenting Model Uncertainty in Deep Learning. In *Proceedings of the 33rd
2579 International Conference on International Conference on Machine Learning
2580 - Volume 48*, ICML’16, pages 1050–1059. JMLR.org, 2016a.
- 2581 Y. Gal and Z. Ghahramani. Bayesian Convolutional Neural Networks with
2582 Bernoulli Approximate Variational Inference, Jan. 2016b. arXiv:1506.02158.

- 2583 Y. Gal and R. Turner. Improving the Gaussian Process Sparse Spectrum Ap-
2584 proximation by Representing Uncertainty in Frequency Inputs. In F. R.
2585 Bach and D. M. Blei, editors, *Proceedings of the 32nd International Con-*
2586 *ference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*,
2587 volume 37 of *JMLR Workshop and Conference Proceedings*, pages 655–664.
2588 JMLR.org, 2015.
- 2589 Y. Gal, J. Hron, and A. Kendall. Concrete Dropout. In I. Guyon, U. V.
2590 Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Gar-
2591 nett, editors, *Advances in Neural Information Processing Systems 30*, pages
2592 3581–3590. Curran Associates, Inc., 2017.
- 2593 J. Geweke and M. Keane. Smoothly mixing regressions. *Journal of Econo-*
2594 *metrics*, 138:252–290, 05 2007.
- 2595 Z. Ghahramani. Bayesian non-parametrics and the probabilistic approach to
2596 modelling. *Philosophical transactions. Series A, Mathematical, physical,*
2597 *and engineering sciences*, 371:20110553, 02 2013.
- 2598 J. Ghosh and R. Ramamoorthi. Bayesian nonparametrics. *Springer Series in*
2599 *Statistics*, 16, 01 2011.
- 2600 E. Gilboa, Y. Saatchi, and J. P. Cunningham. Scaling Multidimensional Infer-
2601 ence for Structured Gaussian Processes. *IEEE Trans. Pattern Anal. Mach.*
2602 *Intell.*, 37(2):424–436, 2015.
- 2603 T. Gneiting. Compactly supported correlation functions. *Journal of Multi-*
2604 *variate Analysis*, 83:493–508, 2002.
- 2605 T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and
2606 estimation. *Journal of the American Statistical Association*, 102:359–378,
2607 2007.
- 2608 G. H. Golub and C. F. Van Loan. *Matrix computations*. The Johns Hopkins
2609 University Press, 3rd edition, Oct. 1996. ISBN 080185413.
- 2610 R. Gramacy. lagp: Large-scale spatial modeling via local approximate gaus-
2611 sian processes in r. *Journal of Statistical Software, Articles*, 72(1):1–46,
2612 2016. ISSN 1548-7660.
- 2613 R. Gramacy and H. Lee. Bayesian treed gaussian process models with an
2614 application to computer modeling. *Journal of the American Statistical As-*
2615 *sociation*, 103, 11 2007.

- 2616 R. B. Gramacy and D. W. Apley. Local gaussian process approximation
2617 for large computer experiments. *Journal of Computational and Graphical*
2618 *Statistics*, 24(2):561–578, 2015.
- 2619 R. B. Gramacy and B. Haaland. Speeding up neighborhood search in local
2620 gaussian process prediction. *Technometrics*, 58(3):294–303, 2016.
- 2621 R. B. Gramacy and H. K. H. Lee. Adaptive design and analysis of supercom-
2622 puter experiments. *Technometrics*, 51(2):130–145, 2009.
- 2623 A. Graves. Practical Variational Inference for Neural Networks. In J. Shawe-
2624 Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger,
2625 editors, *Advances in Neural Information Processing Systems 24*, pages 2348–
2626 2356. Curran Associates, Inc., 2011.
- 2627 C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in
2628 gaussian processes. In *Proceedings of the 22nd International Conference on*
2629 *Machine Learning*, ICML '05, page 265–272, New York, NY, USA, 2005.
2630 Association for Computing Machinery. ISBN 1595931805.
- 2631 C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Mod-
2632 ern Neural Networks. In D. Precup and Y. W. Teh, editors, *Proceedings*
2633 *of the 34th International Conference on Machine Learning*, volume 70 of
2634 *Proceedings of Machine Learning Research*, pages 1321–1330, International
2635 Convention Centre, Sydney, Australia, Aug. 2017. PMLR.
- 2636 D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-
2637 of-distribution examples in neural networks, 2016.
- 2638 J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In
2639 *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*,
2640 pages 282–290. AUAI Press, 2013.
- 2641 J. Hensman, A. Matthews, and Z. Ghahramani. Scalable Variational Gaussian
2642 Process Classification. In *Proceedings of the 18th International Conference*
2643 *on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine*
2644 *Learning Research*, pages 351–360. PMLR, 2015a.
- 2645 J. Hensman, A. G. Matthews, M. Filippone, and Z. Ghahramani. MCMC
2646 for Variationally Sparse Gaussian Processes. In C. Cortes, N. D. Lawrence,
2647 D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Infor-*
2648 *mation Processing Systems 28*, pages 1648–1656. Curran Associates, Inc.,
2649 2015b.

- 2650 G. E. Hinton. Training products of experts by minimizing contrastive diver-
2651 gence. *Neural Comput.*, 14(8):1771–1800, Aug. 2002. ISSN 0899-7667.
- 2652 G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep
2653 belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. ISSN 0899-7667.
- 2654 N. Hjort, C. Holmes, P. Muller, and S. Walker. *Bayesian Nonparametrics*.
2655 Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge
2656 University Press, 2010.
- 2657 Q. M. Hoang, T. N. Hoang, and K. H. Low. A generalized stochastic vari-
2658 ational bayesian hyperparameter learning framework for sparse spectrum
2659 gaussian process regression, 2016.
- 2660 T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine
2661 learning. *Annals of Statistics*, 36(3):1171–1220, 2008.
- 2662 M. Huang, R. Li, H. Wang, and W. Yao. Estimating mixture of gaussian pro-
2663 cesses by kernel smoothing. *Journal of Business and Economic Statistics*,
2664 32, 05 2014.
- 2665 R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixture of local
2666 expert. *Neural Computation*, 3:78–88, 02 1991.
- 2667 M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the
2668 em algorithm. In *Proceedings of 1993 International Conference on Neural
2669 Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pages 1339–1344 vol.2,
2670 1993.
- 2671 A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep
2672 Learning for Computer Vision? In I. Guyon, U. V. Luxburg, S. Bengio,
2673 H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances
2674 in Neural Information Processing Systems 30*, pages 5574–5584. Curran
2675 Associates, Inc., 2017.
- 2676 H. Kim, B. Mallick, and C. Holmes. Analyzing nonstationary spatial data
2677 using piecewise gaussian processes. *Journal of the American Statistical As-
2678 sociation*, 100:653 – 668, 2005.
- 2679 D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In
2680 *3rd International Conference on Learning Representations*, 2015.
- 2681 D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceed-
2682 ings of the Second International Conference on Learning Representations
2683 (ICLR 2014)*, Apr. 2014.

- 2684 D. P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the
2685 Local Reparameterization Trick. In C. Cortes, N. D. Lawrence, D. D. Lee,
2686 M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information
2687 Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.
- 2688 J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process
2689 prediction and observation models. *Autonomous Robots*, 27:75–90, 09 2008.
- 2690 A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with
2691 deep convolutional neural networks. In *Proceedings of the 25th Interna-
2692 tional Conference on Neural Information Processing Systems*, NIPS’12,
2693 pages 1097–1105, USA, 2012. Curran Associates Inc.
- 2694 V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learn-
2695 ing using calibrated regression. In *ICML*, 2018.
- 2696 M. Kull, T. S. Filho, and P. Flach. Beta calibration: a well-founded and easily
2697 implemented improvement on logistic calibration for binary classifiers. In
2698 A. Singh and J. Zhu, editors, *Proceedings of the 20th International Con-
2699 ference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings
2700 of Machine Learning Research*, pages 623–631, Fort Lauderdale, FL, USA,
2701 Apr. 2017. PMLR.
- 2702 B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Pre-
2703 dictive Uncertainty Estimation using Deep Ensembles. In I. Guyon, U. V.
2704 Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Gar-
2705 nett, editors, *Advances in Neural Information Processing Systems 30*, pages
2706 6402–6413. Curran Associates, Inc., 2017.
- 2707 N. D. Lawrence, M. Seeger, and R. Herbrich. Fast Sparse Gaussian Process
2708 Methods: The Informative Vector Machine. In *Advances in Neural Infor-
2709 mation Processing Systems 15*, pages 625–632. MIT Press, 2002.
- 2710 M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R.
2711 Figueiras-Vidal. Sparse Spectrum Gaussian Process Regression. *Journal
2712 of Machine Learning Research*, 11:1865–1881, 2010.
- 2713 Q. Le, T. Sarlos, and A. Smola. Fastfood - Approximating Kernel Expansions
2714 in Loglinear Time. In *30th International Conference on Machine Learning
2715 (ICML)*, 2013.
- 2716 H. Liu, J. Cai, Y. Wang, and Y. Ong. Generalized robust bayesian committee
2717 machine for large-scale gaussian process regression. 07 2018a.

- 2718 H. Liu, Y.-S. Ong, X. Shen, and J. Cai. When gaussian process meets big
2719 data: A review of scalable gps, 2018b.
- 2720 C. Louizos and M. Welling. Structured and efficient variational deep learning
2721 with matrix gaussian posteriors. In *Proceedings of the 33rd International
2722 Conference on International Conference on Machine Learning - Volume 48*,
2723 ICML'16, page 1708–1716. JMLR.org, 2016.
- 2724 C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learn-
2725 ing. In *Advances in Neural Information Processing Systems 30*, pages 3288–
2726 3298. Curran Associates, Inc., 2017.
- 2727 D. MacKay. Introduction to gaussian processes. 1998.
- 2728 D. J. MacKay. Bayesian interpolation. *NEURAL COMPUTATION*, 4:415–
2729 447, 1991.
- 2730 D. J. C. Mackay. *Bayesian Methods for Adaptive Models*. PhD thesis, USA,
2731 1992. UMI Order No. GAX92-32200.
- 2732 D. J. C. Mackay. Bayesian methods for backpropagation networks. In E. Do-
2733 many, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Net-
2734 works III*, chapter 6, pages 211–254. Springer, 1994.
- 2735 J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel
2736 networks. In *Proceedings of the 27th International Conference on Neu-
2737 ral Information Processing Systems - Volume 2*, NIPS'14, page 2627–2635,
2738 Cambridge, MA, USA, 2014. MIT Press.
- 2739 S. Masoudnia and R. Ebrahimpour. Mixture of experts: A literature survey.
2740 *Artificial Intelligence Review*, 42, 08 2014.
- 2741 M. McIntire, D. Ratner, and S. Ermon. Sparse gaussian processes for bayesian
2742 optimization. In *Proceedings of the Thirty-Second Conference on Uncer-
2743 tainty in Artificial Intelligence*, UAI'16, page 517–526, Arlington, Virginia,
2744 USA, 2016. AUAI Press. ISBN 9780996643115.
- 2745 E. Meeds and S. Osindero. An alternative infinite mixture of gaussian process
2746 experts. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances
2747 in Neural Information Processing Systems 18*, pages 883–890. MIT Press,
2748 2006.
- 2749 J. Mercer. Functions of positive and negative type and their connection with
2750 the theory of integral equations. *Proceedings of the Royal Society of London*,
2751 209:415–446, 1909.

- 2752 T. P. Minka. Expectation Propagation for approximate Bayesian inference. In
2753 *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*,
2754 UAI '01, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann
2755 Publishers Inc.
- 2756 D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep
2757 neural networks. In *Proceedings of the 34th International Conference on*
2758 *Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*,
2759 pages 2498–2507. PMLR, 2017.
- 2760 A. H. Murphy. A new vector partition of the probability score. *Journal of*
2761 *Applied Meteorology*, 12(4):595–600, 1973.
- 2762 I. Murray, R. P. Adams, and D. J. C. MacKay. Elliptical slice sampling.
2763 *Journal of Machine Learning Research - Proceedings Track*, 9:541–548, 2010.
- 2764 A. Naish-Guzman and S. Holden. The generalized FITC approximation. In
2765 *Advances in Neural Information Processing Systems 20*, pages 1057–1064.
2766 Curran Associates Inc., 2007. ISBN 978-1-60560-352-0.
- 2767 R. M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statis-*
2768 *tics)*. Springer, 1 edition, Aug. 1996. ISBN 0387947248.
- 2769 T. N. A. Nguyen, A. Bouzerdoum, and S. L. Phung. Variational inference for
2770 infinite mixtures of sparse gaussian processes through kl-correction. In *2016*
2771 *IEEE International Conference on Acoustics, Speech and Signal Processing*
2772 *(ICASSP)*, pages 2579–2583, 2016.
- 2773 T. V. Nguyen and E. V. Bonilla. Automated variational inference for gaussian
2774 process models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence,
2775 and K. Q. Weinberger, editors, *Advances in Neural Information Processing*
2776 *Systems 27*, pages 1404–1412. Curran Associates, Inc., 2014a.
- 2777 T. V. Nguyen and E. V. Bonilla. Fast allocation of gaussian process experts.
2778 In *ICML*, 2014b.
- 2779 A. Niculescu-Mizil and R. Caruana. Predicting Good Probabilities with Su-
2780 pervised Learning. In *Proceedings of the 22Nd International Conference on*
2781 *Machine Learning*, ICML '05, pages 625–632, New York, NY, USA, 2005.
2782 ACM.
- 2783 A. Panos, P. Dellaportas, and M. K. Titsias. Fully Scalable Gaussian Processes
2784 using Subspace Inducing Inputs. 2018. arXiv:1807.02537.

- 2785 C. Park and D. Apley. Patchwork kriging for large-scale gaussian process
2786 regression. *J. Mach. Learn. Res.*, 19(1):269–311, Jan. 2018. ISSN 1532-
2787 4435.
- 2788 C. Park and J. Z. Huang. Efficient computation of gaussian process regression
2789 for large spatial data sets by patching local gaussian processes. *J. Mach.*
2790 *Learn. Res.*, 17(1):6071–6099, Jan. 2016. ISSN 1532-4435.
- 2791 J. Platt. Probabilistic outputs for support vector machines and comparisons
2792 to regularized likelihood methods. *Advances in Large Margin Classifiers*,
2793 10(3), 1999.
- 2794 G. Pleiss, J. Gardner, K. Weinberger, and A. G. Wilson. Constant-time pre-
2795 dictive distributions for Gaussian processes. In *Proceedings of the 35th*
2796 *International Conference on Machine Learning*, volume 80 of *Proceedings*
2797 *of Machine Learning Research*, pages 4114–4123. PMLR, 2018.
- 2798 A. Pourhabib, F. Liang, and Y. Ding. Bayesian site selection for fast Gaussian
2799 process regression. *Institute of Industrial Engineers Transactions*, 46(5):
2800 543–555, 2014a.
- 2801 A. Pourhabib, F. Liang, and Y. Ding. Bayesian site selection for fast gaussian
2802 process regression. *IIE Transactions*, 46(5):543–555, 2014b.
- 2803 M. T. Pratola, H. A. Chipman, J. R. Gattiker, D. M. Higdon, R. McCulloch,
2804 and W. N. Rust. Parallel bayesian additive regression trees, 2013.
- 2805 J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approx-
2806 imate Gaussian process regression. *Journal of Machine Learning Research*,
2807 6:1939–1959, 2005. ISSN 1532-4435.
- 2808 J. R. Quilan. *Decision Trees and Multi-Valued Attributes*, page 305–318. Ox-
2809 ford University Press, Inc., USA, 1988. ISBN 0198537182.
- 2810 A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines.
2811 In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances*
2812 *in Neural Information Processing Systems 20*, pages 1177–1184. Curran
2813 Associates, Inc., 2008.
- 2814 C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process
2815 experts. *Advances in Neural Information Processing Systems*, 2, 04 2002.
- 2816 C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*.
2817 MIT Press, 2006.

- 2818 S. Remes, M. Heinonen, and S. Kaski. Non-stationary spectral kernels. In
2819 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vish-
2820 wanathan, and R. Garnett, editors, *Advances in Neural Information Pro-
2821 cessing Systems 30*, pages 4642–4651. Curran Associates, Inc., 2017.
- 2822 A. Rudi, L. Carratino, and L. Rosasco. FALKON: An Optimal Large Scale
2823 Kernel Method. In *Advances in Neural Information Processing Systems 30*,
2824 pages 3888–3898. Curran Associates, Inc., 2017.
- 2825 W. Rudin. Fourier analysis on groups. 1962.
- 2826 D. Rullière, N. Durrande, F. Bachoc, and C. Chevalier. Nested kriging predic-
2827 tions for datasets with large number of observations. *Statistics and Com-
2828 puting*, 07 2016.
- 2829 S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*.
2830 Pearson Education, 2 edition, 2003.
- 2831 Y. Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD
2832 thesis, University of Cambridge, 2011.
- 2833 H. Salimbeni and M. P. Deisenroth. Doubly stochastic variational inference for
2834 deep gaussian processes. In *Proceedings of the 31st International Conference
2835 on Neural Information Processing Systems, NIPS’17*, page 4591–4602, Red
2836 Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- 2837 M. Seeger and C. Williams. Fast forward selection to speed up sparse Gaussian
2838 process regression. In *Workshop on AI and Statistics 9*. 2003.
- 2839 M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to
2840 speed up sparse Gaussian process regression. In *Artificial Intelligence and
2841 Statistics 9*, 2003.
- 2842 B. Silverman. Some aspects of the spline smoothing approach to non-
2843 parametric regression curve fitting. 1985.
- 2844 A. J. Smola and P. L. Bartlett. Sparse greedy gaussian process regression.
2845 In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural
2846 Information Processing Systems 13*, pages 619–625. MIT Press, 2001.
- 2847 E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-
2848 inputs. In *Advances in Neural Information Processing Systems 18*, pages
2849 1257–1264. MIT Press, 2005.

- 2850 E. Snelson and Z. Ghahramani. Variable noise and dimensionality reduc-
2851 tion for sparse gaussian processes. In *Proceedings of the Twenty-Second*
2852 *Conference on Uncertainty in Artificial Intelligence*, UAI'06, page 461–468,
2853 Arlington, Virginia, USA, 2006. AUAI Press. ISBN 0974903922.
- 2854 E. Snelson and Z. Ghahramani. Local and global sparse Gaussian process ap-
2855 proximations. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh*
2856 *International Conference on Artificial Intelligence and Statistics, AISTATS*
2857 *2007, San Juan, Puerto Rico, March 21-24, 2007*, volume 2 of *JMLR Pro-*
2858 *ceedings*, pages 524–531. JMLR.org, 2007.
- 2859 J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization
2860 of machine learning algorithms. In *Proceedings of the 25th International*
2861 *Conference on Neural Information Processing Systems - Volume 2*, NIPS'12,
2862 page 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.
- 2863 S. Sun and X. Xu. Variational inference for infinite mixtures of gaussian pro-
2864 cesses with applications to traffic flow prediction. *Intelligent Transportation*
2865 *Systems, IEEE Transactions on*, 12:466 – 475, 07 2011.
- 2866 L. S. L. Tan, V. M. H. Ong, D. J. Nott, and A. Jasra. Variational inference
2867 for sparse spectrum gaussian process regression. *Statistics and Computing*,
2868 26(6):1243–1261, Sep 2015. ISSN 1573-1375.
- 2869 M. Titsias and M. Lazaro-Gredilla. Variational inference for mahalanobis dis-
2870 tance metrics in gaussian process regression. In C. J. C. Burges, L. Bottou,
2871 M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in*
2872 *Neural Information Processing Systems 26*, pages 279–287. Curran Asso-
2873 ciates, Inc., 2013.
- 2874 M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian
2875 Processes. In *Proceedings of the 12th International Conference on Artificial*
2876 *Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning*
2877 *Research*, pages 567–574. PMLR, 2009.
- 2878 G.-L. Tran, E. V. Bonilla, J. Cunningham, P. Michiardi, and M. Filippone.
2879 Calibrating deep convolutional gaussian processes. volume 89 of *Proceedings*
2880 *of Machine Learning Research*, pages 1554–1563. PMLR, 16–18 Apr 2019.
- 2881 R. Urtasun and T. Darrell. Sparse probabilistic regression for activity-
2882 independent human pose inference. In *2008 IEEE Conference on Computer*
2883 *Vision and Pattern Recognition*, pages 1–8, 2008.
- 2884 M. van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional Gaussian
2885 Processes, Sept. 2017.

- 2886 S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair. Gaus-
2887 sian process modeling of large scale terrain. In *2009 IEEE International*
2888 *Conference on Robotics and Automation*, pages 1047–1053, 2009.
- 2889 G. Wahba, X. Lin, F. Gao, D. Xiang, R. Klein, and B. Klein. The bias-
2890 variance tradeoff and the randomized gacv. In M. J. Kearns, S. A. Solla, and
2891 D. A. Cohn, editors, *Advances in Neural Information Processing Systems*
2892 *11*, pages 620–626. MIT Press, 1999.
- 2893 M. Welling and Y. W. Teh. Bayesian Learning via Stochastic Gradient
2894 Langevin Dynamics. In L. Getoor and T. Scheffer, editors, *Proceedings*
2895 *of the 28th International Conference on Machine Learning, ICML 2011,*
2896 *Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. Omni-
2897 press, 2011.
- 2898 C. Williams, C. Rasmussen, A. Schwaighofer, and V. Tresp. Observations on
2899 the nyström method for gaussian process prediction. 01 2002.
- 2900 C. K. I. Williams and D. Barber. Bayesian classification with Gaussian pro-
2901 cesses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,
2902 20:1342–1351, 1998.
- 2903 A. Wilson and H. Nickisch. Kernel Interpolation for Scalable Structured Gaus-
2904 sian Processes (KISS-GP). In *Proceedings of the 32nd International Confer-*
2905 *ence on Machine Learning*, volume 37 of *Proceedings of Machine Learning*
2906 *Research*, pages 1775–1784. PMLR, 2015.
- 2907 A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cunningham. Fast kernel
2908 learning for multidimensional pattern extrapolation. In *Proceedings of the*
2909 *27th International Conference on Neural Information Processing Systems*
2910 *- Volume 2, NIPS’14*, page 3626–3634, Cambridge, MA, USA, 2014. MIT
2911 Press.
- 2912 A. G. Wilson, C. Dann, and H. Nickisch. Thoughts on Massively Scalable
2913 Gaussian Processes. 2015. arXiv:1511.01870.
- 2914 A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing. Stochastic Varia-
2915 tional Deep Kernel Learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg,
2916 I. Guyon, and R. Garnett, editors, *Advances in Neural Information Pro-*
2917 *cessing Systems 29*, pages 2586–2594. Curran Associates, Inc., 2016.
- 2918 J. Yang, P. Yu, and B. Kuo. A nonparametric feature extraction and its
2919 application to nearest neighbor classification for hyperspectral image data.
2920 *IEEE Transactions on Geoscience and Remote Sensing*, 48(3):1279–1293,
2921 2010.

- 2922 F. X. Yu, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and
2923 S. Kumar. Orthogonal Random Features. In D. D. Lee, M. Sugiyama,
2924 U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural
2925 Information Processing Systems 29*, pages 1975–1983. Curran Associates,
2926 Inc., 2016.
- 2927 H. Yu, T. N. Hoang, K. H. Low, and P. Jaillet. Stochastic variational inference
2928 for bayesian sparse gaussian process regression, 2017.
- 2929 S. Yuksel, J. Wilson, and P. Gader. Twenty years of mixture of experts. *Neural
2930 Networks and Learning Systems, IEEE Transactions on*, 23:1177–1193, 08
2931 2012a.
- 2932 S. Yuksel, J. Wilson, and P. Gader. Twenty years of mixture of experts. *Neural
2933 Networks and Learning Systems, IEEE Transactions on*, 23:1177–1193, 08
2934 2012b.
- 2935 B. Zadrozny and C. Elkan. Transforming Classifier Scores into Accurate Mul-
2936 ticlass Probability Estimates. In *Proceedings of the Eighth ACM SIGKDD
2937 International Conference on Knowledge Discovery and Data Mining*, KDD
2938 '02, pages 694–699, New York, NY, USA, 2002. ACM.
- 2939 L. Zhao, Z. Chen, and J. Ma. An effective model selection criterion for mix-
2940 tures of gaussian processes. In X. Hu, Y. Xia, Y. Zhang, and D. Zhao, ed-
2941 itors, *Advances in Neural Networks – ISNN 2015*, pages 345–354. Springer
2942 International Publishing, 2015a.
- 2943 L. Zhao, Z. Chen, and J. Ma. An effective model selection criterion for mix-
2944 tures of gaussian processes. In X. Hu, Y. Xia, Y. Zhang, and D. Zhao,
2945 editors, *Advances in Neural Networks – ISNN 2015*, pages 345–354, Cham,
2946 2015b. Springer International Publishing. ISBN 978-3-319-25393-0.

Index

2947

2948 DGP, 4
2949 DTC, 19
2950 FIC, 21
2951 GPs, 3
2952 IPs, 67
2953 KISS-GP, 32
2954 ML, 1
2955 PIC, 22
2956 PITC, 21
2957 RBF, 13
2958 SKI, 33
2959 SOD, 17
2960 SOR, 18

2971

2961 SVGP, 27
2962 SVI, 26
2963 SVM, 16
2964 VSSGP, 38

2965 CNNs, 7
2966 fitc, 20

2967 GPR, 9
2968 GPs, 9

2969 ME, 40

2970 sgpp, 19