Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Data Augmentation in Infinitely Wide Neural Networks

Author:
Yuchen Niu

Supervisor:
Mark van der Wilk

Submitted in partial fulfillment of the requirements for the MSc degree in Advanced Computing of Imperial College London

October 3, 2022

Abstract

Infinitely wide neural networks have shown equivalence to Gaussian processes (GPs). This equivalence enables exact Bayesian inference for the posterior and predictive distribution without ever instantiating a neural network, but by evaluating the corresponding GP. A major advantage of GPs over neural networks is that the hyperparameters can be optimised by backpropagation to maximise the marginal likelihood. However, the evaluation of the marginal likelihood is computationally intractable in large-scale machine learning, especially when the training data is enlarged for better generalisation by generating a broader set of augmentations. In this project, we address the intractability by sparse Gaussian process regression (1) and choose the best model by maximising a variational lower bound of the true log marginal likelihood. We find that the sparse approximation estimates the true posterior on a regression task with few inducing points. However, it shows the difficulty of approximation on classification tasks. A comparison of the efficiency of different methods for selecting inducing points is also conducted. Furthermore, we study the invariances in the data and incorporate them into the model by summing over orbits (2). Empirical results show that the invariances can be learned in a supervised manner and demonstrate different characteristics from the infinitely wide neural networks.

Acknowledgments

I would like to thank my supervisor, Mark van der Wilk, for guiding me through the project. I really enjoyed this project that he proposed, even though it was difficult for me in the first place. His insightful thought and criticism helped me stay on track and push the progress of the project. I would also like to thank my second marker, Yingzhen Li, for her valuable advice in academia and life. Finally, I would like to thank my family and friends for their continuous support.

Contents

Chapter 1

Introduction

1.1 Overview

Neural networks have gained plenty of attention in the machine learning community and achieved remarkable performance on various tasks. Different network architectures are believed to make them well suited to modelling the world. Convolutional neural networks (CNNs) have led to a series of breakthroughs for object recognition and image classification (3, 4). Recurrent neural networks (RNNs) have shown promising results in language modelling and speech recognition due to their ability to handle inputs or outputs with variable lengths (5, 6). More recently, transformers have become the dominating approach in natural language processing and computer vision and it removes the need for task-specific architectures (7, 8, 9).

Neural networks are commonly used in engineering applications due to their ability to fit sophisticated functions. In a feedforward neural network (FFNN), a hidden layer linearly transforms the hidden states, followed by a non-linear activation function. Several works (10, 11, 12) have shown that a multilayer perceptron network with one such hidden layer can approximate any function if a sufficient number of hidden units is used. More complex network architectures, with more layers and hidden units, have advantages in fitting more complex functions. However, they are also prone to overfitting if the training data are limited and biased, which leads to a poor generalisation of unseen data. So, choosing the right amount of layers and units based on background knowledge about the problem and data is often challenging in practice.

Neural networks are "non-parametric" in terms of statistics since there are no apparent parameters that can explain their characteristics. So, it is less interpretable than those of parametric models, such as a single-valued Gaussian process with a mean and variance. Neural networks also demonstrate overconfidence and poor uncertainty estimation (13, 14). These problems limit the usage of neural networks in safety-critical circumstances. In opposite, Bayesian inference is a theoretically principled framework for learning uncertainty. For instance, if the prediction of a Gaussian process on a test sample has a large confidence interval, the prediction of

the test sample can be decided not to be taken until more information is involved.

From a Bayesian perspective, a Bayesian defines a model, selects a prior, and computes the posterior given training data. The model complexity does not depend on the number of data. If the prior and model are selected correctly for a thousand observations, they are correct for a million observations. From the principle of "Occam's Razor", simple models are preferred over complex models when the latter cannot provide benefits for fitting the data. A proper Bayesian approach guarantees the above principle by penalising complex models and using a model we can afford computationally and simple enough to describe the data.

1.2 Goals and Contribution

Given the benefits of Gaussian processes over neural networks, a bridge between them is explored. Neal (15) proves that a single-layered neural network with infinite hidden units is equivalent to a Gaussian process with mild constraints on its prior distribution. Following works (16, 17, 18) extend the equivalences to multilayer neural networks with other common architectures. We will refer to the neural network equivalent Gaussian process as NNGP.

Data augmentation is a simple and efficient method to improve the generalisation of neural networks by enlarging training data size to extrapolate to a large region of the input space correctly. The augmented data are essentially the modified training data in the ways that the modifications do not influence the outputs. Similarly, a well-defined Gaussian process relies on sufficient and reliable data to raise its confidence in test data. However, hand-crafting data relies on expert knowledge and trial and error. It is undesired from Bayesian perspectives according to which assumptions and expert knowledge should be explicitly encoded in the prior since the priors of machine learning applications are often hard to determine.

Gaussian processes face another obstacle: they are computationally intractable on large-scale datasets because the exact inference has complexity $\mathcal{O}(N^3)$ on a dataset of size N. This hinders the application of these networks for many domains in machine learning. So, we aim to solve the intractability by low-rank approximations to the covariance matrices with a series of inducing points. Furthermore, we investigate the data augmentation as learned invariances incorporated in prior distributions and compare it with standard data augmentation. The following list summarises our contribution:

1. Develop the sparse Gaussian process regression for infinitely wide neural networks in JAX[1]

2. Identify the performance impact as varying the number of inducing points and how many inducing points is enough to provide high-quality approximates.

[1]The code is available at GitHub

3. Study how the quality of inducing points affects the convergence of an approximate to the true posterior.
4. Characterise learned invariances and data augmentation, and compare them with the baselines from approximation and generalisation perspectives.

1.3 Outline

This section outlines the structure of the report:

- **Chapter 2** represents the background of the Gaussian processes, infinitely-wide neural networks and data augmentation.
- **Chapter 3** explains in detail the implementation of the sparse Gaussian process regression, inducing points selections and data augmentation.
- **Chapter 4** introduces experiment setups and datasets. The experiment results are also evaluated from approximation and generalisation perspectives.
- **Chapter 5** concludes achievements of the project and reasons the future work.

Chapter 2

Background

2.1 Bayesian Inference

Bayesian inference is known for expressing uncertainty of events, given incomplete information and a preliminary understanding of the world. It uses probability as the subjective state of uncertainty and learning to be more confident in predictions of new samples when more observations are made. Consider a random process that generates a series of data, $D = \{d^{(1)}, d^{(2)}, ..., d^{(N)}\}$, independently from a probabilistic distribution. We can build a probabilistic model with unknown parameters θ to simulate the distribution of D. To estimate θ, one can use maximum likelihood estimation (MLE) given the observed values D:

$$L(\theta|D) \propto p(D|\theta) = \prod_{i=1}^{N} p(d^{(i)}|\theta) \tag{2.1}$$

For instance, in the coin tossing problem, θ is the parameter of a Bernoulli distribution and will be the frequency of heads among observations by MLE. If the number of observations tends to infinity, the estimated θ will converge to the true value. However, MLE does not always work if we only have a finite set of observations. One problem is that the unseen data has zero probability and the likelihood is also zero.

Bayesian learning leverages the Bayes' rule to express a posterior belief regarding the values of parameters, after observing D:

$$p(\theta|D) = \frac{\prod_{i=1}^{N} p(d^{(i)}|\theta)p(\theta)}{\prod_{i=1}^{N} p(d^{(i)})} \propto L(\theta|D)p(\theta) \tag{2.2}$$

The posterior distribution now contains the combined information about θ derived from observations and background knowledge. As more data are observed, the posterior distribution is more concentrated to the true value. The choice of the prior distribution $p(\theta)$ is an essential step since wrong prior can cause slow, or even wrong, convergence. To predict new data $d^{(N+1)}$, Bayesian learning marginalises the model

parameters by integrating the prediction on the new data and the posterior distribution:

$$p(d^{(N+1)}|D) = \int p(d^{(N+1)}|\theta)p(\theta|D)\, d\theta \tag{2.3}$$

In the coin tossing example, the prior distribution is commonly uniform. Assuming that only heads are observed for N tosses, the Bayesian prediction for the next toss given previous tosses will be $p(d^{(N+1)}|D) = [(h+1)/(N+1)$ if $d^{(N+1)} =$ head$; (t+1)/(N+1)$ if $d^{(N+1)} =$ tail$]$, where h and t are the number of heads and tails in the first N tosses, respectively. Unlike the MLE strictly estimates $\hat{\theta} = h/n$, the Bayesian prediction successively converges $\hat{\theta}$ towards the true value as more tosses are observed while allowing some confidence in rare observations.

2.1.1 Gaussian Process

Since a function is defined by a collection of parameters and the parameters are random variables in the Bayesian framework, we can denote the function itself as a random variable if each finite selection of parameters has a Gaussian distribution $\mathcal{N}(f; \mu(\cdot), k(\cdot, \cdot'))$, which its properties are specified by a mean function $\mu(\cdot) = \mathbb{E}[f(\cdot)]$ and a symmetric positive-definite covariance (kernel) function $k(\cdot, \cdot') = cov(f(\cdot), f(\cdot'))$ that take a set of indices or inputs (19). Gaussian processes make the specification of sensible prior distribution easier due to a single random variable and provide better uncertainty estimates by allowing an infinite number of basis functions and kernel tricks.

Another attractive feature of Gaussian processes is an analytic solution for posterior distribution since everything is Gaussian. Given a set of training data $\mathcal{D} = \{(\boldsymbol{X}, \boldsymbol{y})\}$, we can define the prior distribution as $\mathcal{GP}(\mathbf{0}, k(\boldsymbol{X}, \boldsymbol{X}))$ and the likelihood as $p(\boldsymbol{y}|f(\boldsymbol{X}), \boldsymbol{X}) = \mathcal{N}(\boldsymbol{y}; f(\boldsymbol{X}), \sigma^2\boldsymbol{I})$, where $\boldsymbol{y} = f(\boldsymbol{X}) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2)$. Following the Bayes' rule, the posterior of a Gaussian process given the above conditions is:

$$\begin{aligned} p(f(x^*)|\boldsymbol{y}, \boldsymbol{X}, x^*) = \mathcal{N}(f(x^*); k(x^*, \boldsymbol{X})[k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\boldsymbol{I}]^{-1}\boldsymbol{y}, \\ k(x^*, x^*) - k(x^*, \boldsymbol{X})[k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\boldsymbol{I}]^{-1}k(\boldsymbol{X}, x^*)) \end{aligned} \tag{2.4}$$

As we know, we can make predictions with uncertainty estimates given a prior. However, different priors make different predictions of different quality. So, we need to select the right prior for each task which the prior is determined by the hyperparameters, as other unobserved quantities that we can infer with Bayes' rule as Equation 2.2. Since we are looking for a fit that generalises to unseen data, we can maximise the marginal likelihood for Gaussian process regression models:

$$\begin{aligned} \log p(\boldsymbol{y}|\theta, \boldsymbol{X}) &= \log \mathcal{N}(\boldsymbol{y}; \boldsymbol{m}, \boldsymbol{K} + \sigma^2\boldsymbol{I}) \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{K} + \sigma^2\boldsymbol{I}| - \frac{1}{2}(\boldsymbol{y} - \boldsymbol{m})^\top(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{m}) \end{aligned} \tag{2.5}$$

which the log determinant term penalises the model complexity and the quadratic term measures whether observations are within the variations allowed by the prior.

Practically, we often encounter intractable integral in Bayesian inference since analytical tractability is only available for conjugate likelihoods. For the binary classification tasks, the likelihood should be the Bernoulli distribution. There is no closed-form solution for the marginal likelihood $p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{X}, \theta)P(\theta)\, d\theta$, so as for the predictive distribution. Thus, we need Gaussian approximations to evaluate the intractable integrals.

2.1.2 Monte Carlo Methods

Monte Carlo (MC) approximation has been used for years in statistical physics. It is also applied in statistics to solve intractable computations. Most integrals in the Bayesian framework are in fact expectations. For example, the prediction of a model with parameters θ can be written as

$$\begin{aligned} p(y^*|x^*, \boldsymbol{y}, \boldsymbol{X}, \theta) &= \int p(y^*|x^*, \theta)P(\theta|\boldsymbol{y}, \boldsymbol{X})\, d\theta \\ &= \mathbb{E}_{p(\theta|\boldsymbol{y},\boldsymbol{X})}[p(y^*|x^*, \theta)] \end{aligned} \tag{2.6}$$

Any expectation can be generalised as $\mathbb{E}_{p(x)}[g(x)] \approx \frac{1}{S}\sum_{s=1}^{S} g(x^{(s)})$ with $x^{(s)} \overset{\text{iid}}{\sim} p(x)$, where samples $x^{(s)}$ are generated from a distribution of interest $p(x)$. Nevertheless, not all distributions can be directly sampled.

Reject sampling overcomes the difficulty of sampling from the posterior distribution $p(x)$ by evaluating the unnormalised distribution (i.e., the numerator in Equation 2.2). The prior controls the generation of candidate networks, and the likelihood controls which candidates are accepted. However, reject sampling is not efficient in high dimensions since its reject rate is very high. Instead of naively omitting rejected samples, importance sampling weights the samples according to the ratio between $p(x)$ and proposal distribution $q(x)$ by rewriting the integral as the expectation under $q(x)$: $\mathbb{E}_{p(x)}[g(x)] = \mathbb{E}_{q(x)}[g(x)\frac{p(x)}{q(x)}]$. However, it still requires many draws from proposal density $q(x)$ to reduce the variance of estimations in high dimensions.

Unlike previous samples generated independently, Markov Chain Monte Carlo (MCMC) makes the Markov assumption that a proposal density $q(x)$ depends and only depends on the previous sample $x^{(t-1)}$. So, a sequence of samples is generated with a joint density

$$q(x^{(1)}, x^{(2)}, ..., x^{(N)}) = q(x^{(1)}) \prod_{t=2}^{N} T(x^{(t)}|x^{(t-1)}) \tag{2.7}$$

where T is the transition distribution for a new state to follow the current state. To ensure a Markov Chain converges to the desired equilibrium distribution $p(x)$, the MCMC must satisfy two properties. First, each step must leave the distribution invariant, $q(x') = \int T(x'|x)q(x)\, dx$, which is implied by a stronger sufficient condition of detailed balance $T(x'|x)q(x) = T(x|x')q(x')$ for all x and x' in the chain. Second, the transition must be ergodic. Thus, there is a unique equilibrium distribution from any initial state. The main disadvantage of MCMC is that it may take a long time to converge.

2.1.3 Variational Inference

Variational inference is the most scalable approximate inference method so far that can overcome the deficiency of previous Monte Carlo methods, which takes many draws from the proposal distribution to minimise the variance of unbiased estimates. Instead, variational inference approximates a probabilistic distribution by optimising an objective function with respect to variational parameters that define an approximating distribution. Since the estimates in the variational inference are biased, we try to minimise the bias of estimates by optimisation.

In variational inference, we wish to define a lower bound that the posterior recovers the marginal likelihood

$$\begin{aligned}
\log p(x) &= \log \int p(x|z)p(z)\,dz \\
&= \log \int p(x|z)\frac{p(z)}{q(z)}q(z)\,dz \\
&= \log \mathbb{E}_q\left[p(x|z)\frac{p(z)}{q(z)}\right] \\
&\geq \mathbb{E}_q \log\left(p(x|z)\frac{p(z)}{q(z)}\right) \quad \text{(Jensen's inequality)} \\
&= \mathbb{E}_q\left[\log p(x|z)\right] - \mathbf{KL}[q(z)||p(z)] = \mathcal{L}(q(z))
\end{aligned} \tag{2.8}$$

This lower bound is also called Evidence Lower Bound (ELBO) and we can optimise the bound for an approximate $q(z)$. The terms in the ELBO only include prior and likelihood that can evaluate and are often integral in close form. This objective function also allows the comparison between different approximations.

Similar to other objective functions, like mean squared errors, which measure the distance between two points, the difference between the log marginal likelihood and the ELBO indicates the similarity of an approximate and the true posterior

$$\begin{aligned}
\log p(x) - \mathcal{L}(q(z)) &= \log p(x) - \int q(z)\log\frac{p(x|z)p(z)}{q(z)}\,dz \\
&= \int q(z)\log p(x)\,dz - \int q(z)\log\frac{p(x|z)p(z)}{q(z)}\,dz \\
&= \int q(z)\log\frac{q(z)}{p(z|x)}\,dz \\
&= \mathbf{KL}[q(z)||p(z|x)]
\end{aligned} \tag{2.9}$$

Therefore, maximising the ELBO is equal to minimising the KL-divergence between an approximate and the true posterior. Moreover, suppose we split the ELBO into two terms. In that case, the expectation term represents the data fit which measures how well the samples from $q(z)$ explain the data and places the mass of $q(z)$ on its MAP estimate; the KL-divergence term represents a regulariser that $q(z)$ should not differ much from its prior $p(z)$.

2.2 Infinitely Wide Neural Networks

Neal (15) shows that the Central Limit Theorem implies that the function computed by a single-layer feedforward neural network in the limit of infinite width is a function drawn from a Gaussian process. But, a problem with conducting Bayesian inference on a neural network is that the prior weights and biases are hard to determine to express our beliefs. The past work (20, 21, 22) shows that criteria for selecting a suitable prior can be found without a complete understanding of the tasks. Then, a prior with zero means and bounded variances is proposed to address this problem for single-layer, infinitely-wide neural networks. The prior ensures that a network converges to a Gaussian process as its width approaches the infinite limit.

Defining a single-layer neural network with H hidden units which takes real-valued inputs $\boldsymbol{x}$. The ith component of network outputs is written as

$$f_i^1(\boldsymbol{x}) = \sum_{j=1}^{H} \boldsymbol{W_{ij}^1 h_j(x)} + \boldsymbol{b_i^1} \tag{2.10}$$

$$h_j^1(\boldsymbol{x}) = \phi(\sum_{k=1}^{d_{in}} \boldsymbol{W_{jk}^0 x_k} + \boldsymbol{b_j^0}) \tag{2.11}$$

where $\phi(\cdot)$ is a nonlinear activation function, $\boldsymbol{W^l}$ and $\boldsymbol{b^l}$ are the weights and bias of layer l, respectively. Since weights and biases are drawn independently from a Gaussian prior, the hidden states after activation (Equation 2.11) are mutually independent. As shown in Equation 2.10, $f_i^1(\boldsymbol{x})$ is the sum of independent and identical distributed (i.i.d.) hidden states. So, the output $f_i^1(\boldsymbol{x})$ follows the Central Limit Theorem that will be Gaussian distributed in the limit of infinite width $H \to \infty$. To obtain a well-defined limit for the prior distribution in such case at any point, the prior variance of the hidden-to-output weights needs to be scaled by the number of hidden units $\sigma_w = \hat{\sigma}_w / H$, where the $\hat{\sigma}_w$ remains fixed. Therefore, the joint distribution of the outputs for multiple inputs converges to a multivariate Gaussian and $f_i^1(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, in which the covariance $\boldsymbol{K}$ is defined by a kernel function

$$k(\boldsymbol{x}, \boldsymbol{x'}) \equiv \mathbb{E}[f_i^1(\boldsymbol{x}), f_i^1(\boldsymbol{x'})] = \sigma_b^2 + \sigma_w^2 \mathbb{E}[h_i^1(\boldsymbol{x}), h_i^1(\boldsymbol{x'})] \equiv \sigma_b^2 + \sigma_w^2 C(\boldsymbol{x}, \boldsymbol{x'}) \tag{2.12}$$

where $C(\boldsymbol{x}, \boldsymbol{x'})$ is introduced in Neal (15).

Lee et al. (16) extend the single layer case to deeper layers by induction. Suppose that the activations of the previous layer are independently and identically Gaussian distributed. The outputs of the current layer are still a sum of i.i.d. terms. So, as the width of the current layer tends to infinity, the joint distribution of outputs forms a multivariate Gaussian distribution with zero mean and covariance that depends on the post-activations of values from the previous layer

$$\begin{aligned} k^l(\boldsymbol{x}, \boldsymbol{x'}) &= \sigma_b^2 + \sigma_w^2 \mathbb{E}_{\boldsymbol{f_i^{l-1}} \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{K^{l-1}})}[\phi(f_i^{l-1}(\boldsymbol{x})), \phi(f_i^{l-1}(\boldsymbol{x'}))] \\ &= \sigma_b^2 + \sigma_w^2 F_\phi[k^{l-1}(\boldsymbol{x}, \boldsymbol{x'})), k^{l-1}(\boldsymbol{x}, \boldsymbol{x}), k^{l-1}(\boldsymbol{x'}), \boldsymbol{x'}))] \quad \text{(by induction)} \end{aligned} \tag{2.13}$$

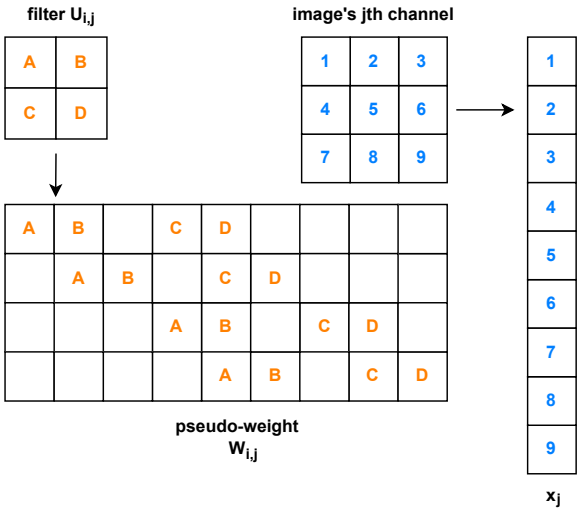


Figure 2.1: Transformation from 2D convolution $\boldsymbol{U_{i,j}} * \boldsymbol{X_j}$ to dot product $\boldsymbol{W_{i,j}}\boldsymbol{x_j}$.

This gives an iterative computation which can obtain $\boldsymbol{K^L}$ for describing the network's final output. To avoid prohibitively expensive integrals for each pair of samples and layers, they improve the cost of kernel computation by storing the values of the function F_ϕ in a lookup table. In their work, the comparison of GP analogue to neural networks optimised by stochastic gradient descent (SGD) is suggestive that the optimization method mimics Bayesian inference.

Another concurrent work (23) overlaps some contributions described in the previous paragraph. In their work, linearly bounded functions solve the heavy tail behaviour induced by nonlinearities. This property is embedded in commonly used nonlinear functions, such as ReLU and ELU. By adapting the kernel from Cho and Saul (24) for nonlinearities, they also derive the convergence of a Bayesian neural network to a Gaussian distribution in a recurrent manner. Unlike Lee et al. (16), which sequentially takes the number of units in each layer to be infinite, the layers grow simultaneously at different rates, which is more practical.

Recent studies have applied the idea of infinite width to other network architectures. Convolutional neural networks (CNNs) have powerful pattern recognition capability in tasks like image classification and face recognition. A convolutional layer takes an arbitrary 2-dimensional input $\boldsymbol{X_j}$ at channel j and multiplies a segment of $\boldsymbol{X_j}$ with a learnable filter $\boldsymbol{U_{i,j}}$ to form the element i of an activation. This computation can be seen as a linear transformation of linearised $\boldsymbol{x_j}$ by a pseudo-weight matrix $\boldsymbol{W_{i,j}}$, as shown in Figure 2.1. Formally, we can write a convolutional neural network as following

$$
\begin{aligned}
f_i^1(\boldsymbol{x}) &= \sum_{j=1}^{C^{(0)}} \boldsymbol{W_{ij}^{(1)}}\boldsymbol{x_j} + \boldsymbol{b_i^1}\mathbf{1} \\
f_i^{(l+1)}(\boldsymbol{x}) &= \sum_{j=1}^{C^{(l)}} \boldsymbol{W_{ij}^{(l+1)}}\phi(f_j^{(l)}(\boldsymbol{x})) + \boldsymbol{b_i^{(l+1)}}\mathbf{1}
\end{aligned}
\tag{2.14}
$$

where $C^{(l)}$ is the total number of channels of the input at layer l. Therefore, we can induce an equivalent Gaussian process for a convolutional neural network like feedforward neural networks.

Following the proofs by Lee et al. (16) and Matthews et al. (23), Garriga-Alonso et al. (25) and Novak et al. (17) derive the same kernel function as Equation 2.13 for the Gaussian process analogue of a CNN in the limit of infinite filters. Instead of computing the covariance of activations between all pairs of locations in the feature map, the covariance of activations $\mathbb{E}_{\boldsymbol{f}_i^{l-1} \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{K}^{l-1})}[\phi(f_i^{l-1}(\boldsymbol{x})), \phi(f_i^{l-1}(\boldsymbol{x'}))]$ is only considered as a diagonal matrix in the CNN since the channels are i.i.d. conditioned on activations from the previous layer. However, they demonstrate that the Gaussian processes cannot outperform the parametric convolutional neural networks.

Yang (18) further derives that the Gaussian process analogue on other neural network components, such as layer normalisation, (gated) recurrent neural networks (RNNs) and attention. The induction in the previous proofs (16) breaks down in some architectures like RNNs since the weight matrix is shared and is no longer conditionally an i.i.d. random Gaussian matrix. The Gaussian conditioning technique addresses this situation. Under a slightly different restriction of nonlinearity, they prove the convergence to a Gaussian process correspondence from all standard architectures with infinite width.

In practice, Gaussian inference often encounters non-conjugate likelihood and prohibitive matrix computation, especially for large datasets, since the predictive distribution depends on integrating the posterior distribution (Equation 2.3). The common solution to address this problem is Monte Carlo estimates. Neal (15) uses the Markov Chain Monte Carlo method based on a hybrid Monte Carlo algorithm, which consists of a variation of the Metropolis algorithm and Gibbs sampling. The following work (23, 17) also leverages Monte Carlo methods to estimate the posterior and predictive distributions. In this project, we address the problem by optimising a variational lower bound, using a feasible amount of computation time and space on a single machine.

2.3 Data Augmentation

Data augmentation is a simple and efficient method to make neural networks have better generalisation by altering data in ways irrelevant to the prediction. Images can be flipped, rotated and scaled (26). However, standard data augmentation produces only limited plausible data. The recent success of deep generative models makes the data augmentation learnable in an unsupervised manner. Antoniou et al. (27) takes an arbitrary image from a source domain and generates other within-class data without the dependency on the classes. In natural language processing, a text can be modified by synonym replacement, random insertion, random swap or random deletion while keeping its semantic representation (28). Leveraging the inconsistent syntactic transformation of machine translation, a sentence can be paraphrased by back translation from other languages (29, 30).

Chapter 3

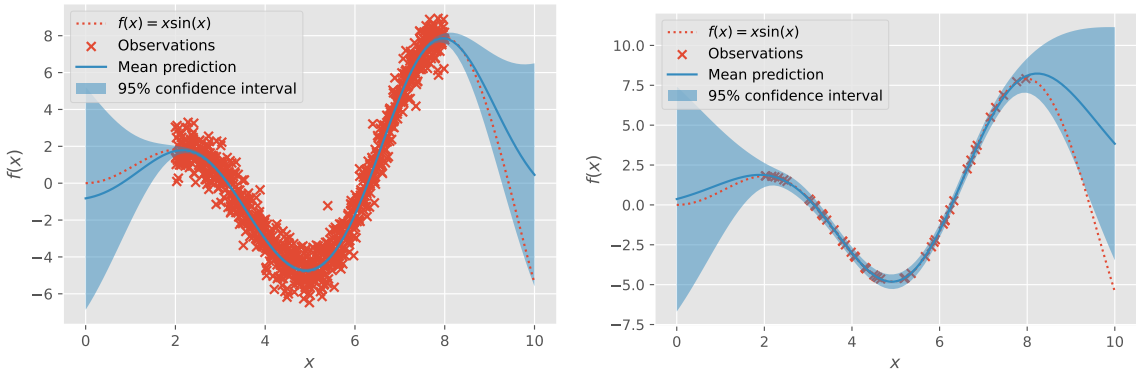
Implementation

3.1 Sparse Gaussian Process Approximations

The Gaussian framework does not only predict unseen data, it also estimates the uncertainties of predictions for better decision making. Although we have discussed the analytic intractability of non-conjugate likelihood and its countermeasures in the section 2.1.2 and 2.1.3, the Gaussian inference can be computationally intractable even for a moderately sized dataset. From a modelling point of view, people are often interested in the marginal likelihood (Equation 2.5) and the predictive distribution (Equation 2.3), which indicates the "training performance" and probabilistic forecasting, respectively. If the prior and likelihood distributions are Gaussian, these distributions of interest can be derived analytically in closed form. However, they and their derivatives include the inverse of a large covariance matrix, or decomposition of the covariance matrix for stability and fast computation, which takes $\mathcal{O}(N^3)$ operations for a dataset of size N.

Because the observations we collected from nature are noisy and are often clustered in a region with similar function values, much redundant information is brought into the Bayesian framework to represent the distribution of the latent function (i.e., the GP posterior). However, we can utilise the strong constraints on the neighbouring outputs around the observations from the GP prior. Suppose we have a GP prior with very high uncertainties everywhere, and we only observe a single data. In that case, this point is learned very well, and the uncertainties of nearby points also decrease. So, the distribution in a region can be represented by the information from observation and prior constraints on neighbouring points. This situation is also illustrated in Figure 3.1. The mean predictions and confidence intervals on these Gaussian processes are almost identical, even though Figure 3.1(b) only uses 5% of training data compared to Figure 3.1(a).

Following the above intuition, inducing point approximations are introduced. This report will revisit the details of the sparse Gaussian process regression (SGPR) (1). Compared to its predecessors that view the inducing point approximations as likelihood approximation (31) and model approximations (32, 33), SGPR estimates



(a) (b)

Figure 3.1: The comparison between Gaussian process regressions on a noisy dataset and noise-free dataset. The target function in the figures is $f(x) = x \times sin(x)$. (a) The 1000 noisy observations are sampled between 2 and 8 with a noise scale of 0.75. (b) 50 noise-free observations are selected randomly between 2 and 8.

the posterior directly by optimising the KL divergence between the approximation and the true posterior. We start by defining the function values of the data $\boldsymbol{X}$ as $\boldsymbol{f} = f(\boldsymbol{X})$ and the function values of inducing points $\boldsymbol{Z}$ as $\boldsymbol{u} = f(\boldsymbol{Z})$. Since a Gaussian process is essentially a Gaussian distribution defined through two real-valued functions (19), a joint multivariate Gaussian distribution can be defined between a Gaussian process and inducing points

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{u} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu_f} \\ \boldsymbol{\mu_u} \end{pmatrix}, \begin{pmatrix} \boldsymbol{K_{ff}} & \boldsymbol{K_{uf}} \\ \boldsymbol{K_{fu}} & \boldsymbol{K_{uu}} \end{pmatrix} \right) \tag{3.1}$$

This equation proposes a degenerate distribution over $\boldsymbol{f}$ and $\boldsymbol{u}$ because $\boldsymbol{u}$ is the GP evaluation at the inducing points and is entirely determined by the latent function $f(\cdot)$ (34).

By the Gaussian conditioning formula and assuming that there is another Gaussian distribution $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{m_u}, \boldsymbol{S_{uu}})$, the conditional distribution of $f(\cdot)$ given $\boldsymbol{u}$ is derived

$$p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{\mu_f} + \boldsymbol{K_{fu}}\boldsymbol{K_{uu}^{-1}}(\boldsymbol{u} - \boldsymbol{\mu_u}), \boldsymbol{K_{ff}} - \boldsymbol{K_{fu}}\boldsymbol{K_{uu}^{-1}}\boldsymbol{K_{uf}}) \tag{3.2}$$

From the above equation, both terms are computed in $\mathcal{O}(NM^2)$, where N is the data size and M is the number of inducing points. Since we are interested in the approximate of the posterior distribution, $\boldsymbol{u}$ should be marginalised like hyperparameters, $q(\boldsymbol{f}) = \int p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u})\, d\boldsymbol{u}$. The resulting marginal distribution is again Gaussian

$$\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu_f} + \boldsymbol{K_{fu}}\boldsymbol{K_{uu}^{-1}}(\boldsymbol{m_u} - \boldsymbol{\mu_u}), \boldsymbol{K_{ff}} - \boldsymbol{K_{fu}}\boldsymbol{K_{uu}^{-1}}(\boldsymbol{K_{uu}} - \boldsymbol{S_{uu}})\boldsymbol{K_{uu}^{-1}}\boldsymbol{K_{uf}}) \tag{3.3}$$

Therefore, the above equation defines the sparse GP posterior (1) which is still computed in $\mathcal{O}(NM^2)$ and the expensive inverse of the full kernel matrix is no longer required.

The question that now arises is how to select the best model. The model selection is based on maximising the marginal likelihood for exact Gaussian processes. In SGPR,

the model can be selected rigorously by minimising the KL divergence between the approximated posterior $q(\boldsymbol{f})$ and the true posterior $p(\boldsymbol{f}|\boldsymbol{X})$. Since minimising the KL divergence between these distributions is equivalent to maximising the variational bound (35), Titsias (1) introduces a variational lower bound for SGPR

$$\mathcal{L} = \log \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{Q_{ff}} + \sigma^2 \boldsymbol{I}) - \frac{1}{2}\sigma^{-2} Tr(\boldsymbol{K_{ff}} - \boldsymbol{Q_{ff}}) \tag{3.4}$$

where $\boldsymbol{Q_{ff}} = \boldsymbol{K_{fu}}\boldsymbol{K_{uu}^{-1}}\boldsymbol{K_{uf}}$. The trace term represents the total variance of the conditional prior $p(\boldsymbol{f}|\boldsymbol{u})$ that measures the squared error of the latent values $\boldsymbol{f}$ from the inducing outputs $\boldsymbol{u}$. If the trace term equals zero, the Nyström approximation is exact and the exact GP prediction can be recovered.

To obtain an efficient and stable evaluation on the lower bound $\mathcal{L}$, we first rewrite the inverse of the covariance matrix by Woodbury identity

$$[\boldsymbol{Q_{ff}} + \sigma^2 \boldsymbol{I}]^{-1} = \sigma^{-2}\boldsymbol{I} - \sigma^{-4}\boldsymbol{K_{fu}}[\boldsymbol{K_{uu}} + \boldsymbol{K_{uf}}\boldsymbol{K_{fu}}\sigma^{-2}]^{-1}\boldsymbol{K_{uf}} \tag{3.5}$$

This only requires a single inverse operation on a $M \times M$ matrix instead of two inverse operations in Equation 3.4. Then, the Cholesky decomposition is applied on the kernel matrix $\boldsymbol{K_{uu}} = \boldsymbol{L}\boldsymbol{L}^\top$ for faster determinant and inverse calculation

$$\begin{aligned}[\boldsymbol{Q_{ff}} + \sigma^2 \boldsymbol{I}]^{-1} &= \sigma^{-2}\boldsymbol{I} - \sigma^{-4}\boldsymbol{K_{fu}}\boldsymbol{L}^{-\top}\boldsymbol{L}^\top[\boldsymbol{L}\boldsymbol{L}^\top + \boldsymbol{K_{uf}}\boldsymbol{K_{fu}}\sigma^{-2}]^{-1}\boldsymbol{L}\boldsymbol{L^{-1}}\boldsymbol{K_{uf}} \\ &= \sigma^{-2}\boldsymbol{I} - \sigma^{-4}\boldsymbol{K_{fu}}\boldsymbol{L}^{-\top}[\boldsymbol{I} + \boldsymbol{L^{-1}}\boldsymbol{K_{uf}}\boldsymbol{K_{fu}}\boldsymbol{L}^{-\top}\sigma^{-2}]^{-1}\boldsymbol{L^{-1}}\boldsymbol{K_{uf}}\end{aligned} \tag{3.6}$$

The above matrix is better conditioned since many kernels are bounded by eigenvalues (19). To simplify this equation, we will denote $\boldsymbol{A} = \boldsymbol{L^{-1}}\boldsymbol{K_{uf}}\sigma^{-1}$ and $\boldsymbol{B} = \boldsymbol{I} + \boldsymbol{A}\boldsymbol{A}^\top$. So, the inverse matrix is now derived as

$$[\boldsymbol{Q_{ff}} + \sigma^2 \boldsymbol{I}]^{-1} = \sigma^{-2}\boldsymbol{I} - \sigma^{-2}\boldsymbol{A}^\top\boldsymbol{B^{-1}}\boldsymbol{A} \tag{3.7}$$

Similar steps are also applied to the determinant term, in addition to the matrix determinant lemma

$$\begin{aligned}|\boldsymbol{Q_{ff}} + \sigma^2 \boldsymbol{I}| &= |\boldsymbol{K_{uu}} + \boldsymbol{K_{uf}}\boldsymbol{K_{fu}}\sigma^{-2}||\boldsymbol{K_{uu}^{-1}}||\sigma^2\boldsymbol{I}| \\ &= |\boldsymbol{L}\boldsymbol{L}^\top + \boldsymbol{K_{uf}}\boldsymbol{K_{fu}}\sigma^{-2}||\boldsymbol{L}^{-\top}||\boldsymbol{L^{-1}}||\sigma^2\boldsymbol{I}| \\ &= |\boldsymbol{I} + \boldsymbol{L^{-1}}\boldsymbol{K_{uf}}\boldsymbol{K_{fu}}\boldsymbol{L}^{-\top}\sigma^{-2}||\sigma^2\boldsymbol{I}| \\ &= |\boldsymbol{B}||\sigma^2\boldsymbol{I}|\end{aligned} \tag{3.8}$$

However, the determinant $|\boldsymbol{B}|$ takes $\mathcal{O}(M!)$ operations. Since $\boldsymbol{B}$ can be decomposed by the Cholesky decomposition and the determinant of a triangular matrix is the product of its diagonal, the determinant term is simplified as

$$|\boldsymbol{B}| = |\boldsymbol{L_B}|^2 = \left(\prod_{i=1}^{M} \boldsymbol{L_{B_{ii}}}\right)^2 \tag{3.9}$$

where the Cholesky decomposition takes $\mathcal{O}(M^3)$ and the computation of determinant only takes $\mathcal{O}(M)$. As a result, the complexity reduces to $\mathcal{O}(M^3)$.

With the above definitions, the variational lower bound of SGPR is expanded as

$$\begin{aligned}\mathcal{L} = -\frac{N}{2}\log 2\pi - \frac{N}{2}\log\sigma^2 - \log\sum_{i=1}^{M}\boldsymbol{L_{B_{ii}}} - \frac{1}{2}\sigma^{-2}\boldsymbol{y}^\top\boldsymbol{y} + \frac{1}{2}\boldsymbol{c}^\top\boldsymbol{c} \\ -\frac{1}{2}\sigma^{-2}Tr(\boldsymbol{K_{ff}}) + \frac{1}{2}Tr(\boldsymbol{A}\boldsymbol{A}^\top)\end{aligned} \tag{3.10}$$

where $\boldsymbol{c} = \boldsymbol{L_B^{-1}}\boldsymbol{A}\boldsymbol{y}\sigma^{-1}$ and $\boldsymbol{c}^\top\boldsymbol{c} = \sigma^{-2}\boldsymbol{y}^\top\boldsymbol{A}^\top\boldsymbol{B^{-1}}\boldsymbol{A}\boldsymbol{y}$. Practically, we also need to consider the case of multivariate outputs. Since each output is conditionally independent given a model and data, we can simply sum the ELBOs of conditionally independent GPs. So, we obtain the ELBO in general that is computationally tractable ($\mathcal{O}(NM^2)$) and optimisable.

Even though the model selection is feasible by picking the higher ELBO values, the distance between the ELBO and the exact marginal likelihood is still unknown. In order to measure how well the sparse method approximates the true posterior, Titsias (36) constructs an upper bound (EUBO) for the special case of Gaussian process regression

$$\mathcal{U} = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{Q_{ff}} + \sigma^2\boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^\top(\boldsymbol{Q_{ff}} + (t+\sigma^2)\boldsymbol{I})^{-1}\boldsymbol{y} \tag{3.11}$$

where $t = Tr(\boldsymbol{K_{ff}} - \boldsymbol{Q_{ff}}) \geq \lambda_{\max}(\boldsymbol{K_{ff}} - \boldsymbol{Q_{ff}})$ and it compensates for the smaller eigenvalues of $\boldsymbol{Q_{ff}}$ such that the quadratic term is over-estimated. We follow similar steps as the ELBO to expand the EUBO

$$\mathcal{U} = -\frac{N}{2}\log 2\pi - \frac{N}{2}\log\sigma^2 - \log\sum_{i=1}^{M}\boldsymbol{L_{B_{ii}}} - \frac{1}{2}\alpha^{-1}\boldsymbol{y}^\top\boldsymbol{y} + \frac{1}{2}\boldsymbol{c'}^\top\boldsymbol{c'} \tag{3.12}$$

where $\boldsymbol{c'} = \boldsymbol{L_{B'}^{-1}}\boldsymbol{A'}\boldsymbol{y}\alpha^{-1/2}$. The matrices in the term $\boldsymbol{c'}$ are replaced by $\boldsymbol{A'} = \boldsymbol{L^{-1}}\boldsymbol{K_{uf}}\alpha^{-1/2}$ and $\boldsymbol{B'} = \boldsymbol{I} + \boldsymbol{A'}\boldsymbol{A'}^\top$. α is the "corrected noise" $t + \sigma^2$ in the inverse matrix.

For prediction, we need to infer the mean and covariance of the variational approximation for a new point. Because the information in the approximated posterior is from the Gaussian prior $q(\boldsymbol{u})$ (37), we conduct the same integration over the inducing variables $\boldsymbol{u}$ as Equation 3.3 for the approximated distribution of f^*

$$f^* \sim \mathcal{N}(\boldsymbol{K_{*u}}\boldsymbol{K_{uu}^{-1}}\boldsymbol{m_u}, \boldsymbol{K_{**}} - \boldsymbol{K_{*u}}\boldsymbol{K_{uu}^{-1}}(\boldsymbol{K_{uu}} - \boldsymbol{S_{uu}})\boldsymbol{K_{uu}^{-1}}\boldsymbol{K_{u*}}) \tag{3.13}$$

We omit μs in the above equation since zero-mean Gaussian outputs are assumed. For most tasks in machine learning, outputs can be preprocessed to have zero mean.

In this project, we implement the ELBO (Equation 3.10), EUBO (Equation 3.12) and predictive distribution (Equation 3.13) in JAX in the context of infinitely-wide neural networks. We will refer to our method as sparse neural network equivalent Gaussian processes (SNNGP) for simplicity. Our design is inspired by the GPflow (38, 39). We rebuild the "wheel" since Neural Tangents (40) is implemented in JAX, and

the automatic differentiation does not support the propagation between JAX and Tensorflow.

Concurrent work on the sparse approximation in JAX is also conducted by GPJax (41). The optimisations in GPJax allow JAX built-in, first-order optimisers and BFGS. However, first-order optimisers (e.g., Adam (42)) may converge slowly compared with second-order methods, and BFGS is not appropriate for large-scale problems due to the expensive Hessian approximation. In contrast, our implementation supports the optimisation of the hyperparameters through Limited-memory BFGS (L-BFGS) algorithm (43). The L-BFGS is a quasi-Newton method that approximates the BFGS algorithm with a limited amount of past updates. Moreover, the sparse approximations in the GPJax optimise the locations of inducing points. We revoke the optimisation on inducing locations for better memory efficiency, which is presented in the next section.

3.2 Inducing Points Selection

In this project, we experiment with two strategies for selecting inducing points. The first method is randomly selecting inducing points from training samples. However, this method introduces uncertainties on the quality of inducing points. Randomly selected inducing points may neglect important locations or cause redundant information. The poor choice of inducing points is not a significant problem in the original SGPR (1) since the locations of inducing points are optimised as hyperparameters. However, as the scale of the dataset increases, optimising model parameters and inducing points simultaneously are challenging in terms of computational complexity and optimisation dynamics. Therefore, in this project, we only consider optimising model parameters with fixed inducing points.

The second method is fast greedy maximum a posteriori (MAP) inference (44). This algorithm is initially designed for determinantal point process (DPP) (45) that is a probabilistic model with applications in machine learning tasks like summarisation (46, 47) and search (48). A characteristic of DPP is that it assigns a higher probability to subsets of items which are diverse from each other (49). By incrementally updating the Cholesky factor, fast greedy MAP inference performs exact implementation of a greedy algorithm (50) with great acceleration. It reduces the complexity from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$ and runs in $\mathcal{O}(NM^2)$ time to return M values from N items, without scarifies accuracy. For the rest of the report, we will refer to fast greedy MAP inference for inducing points selection as greedy selection.

The study of greedy selection on sparse Gaussian processes is also conducted by Burt et al. (51). They show that greedy selection with a slightly larger number of inducing points can obtain the same performance as optimised inducing points. Motivated by their finding, we implement this method in JAX for a more diverse selection of inducing points. Our implementation details are shown in Algorithm 1. Since the computation of variances (diagonal of the kernel matrix) relies on the model parameters, Burt et al. suggest that the selection of inducing points and the

Algorithm 1 Fast Greedy MAP Inference for Inducing Points Selection

Input: Training data $\boldsymbol{X}$, number of inducing points M, kernel function $k(\cdot, \cdot')$

$\boldsymbol{c_i} = zeros[M, N]$
indices $= zeros[M] + N$
$\boldsymbol{d} = diag(k(\boldsymbol{X}, \boldsymbol{X}))$
indices$[0] = \arg\max(\boldsymbol{d})$
$m = 0$
while $m < M - 1$ **do**
    $j =$ indices$[m]$
    $d_j = \sqrt{\boldsymbol{d}[j]}$
    $\boldsymbol{c_j} = \boldsymbol{c}_i[: m, j]$
    $\boldsymbol{L} = k(X, X[j])$
    $\boldsymbol{e_i} = (\boldsymbol{L} - \langle \boldsymbol{c}_j, \boldsymbol{c_i}[: m] \rangle)/d_j$
    $\boldsymbol{c_i}[m, :] = \boldsymbol{e_i}$
    $\boldsymbol{d} = \boldsymbol{d} - \boldsymbol{e_i^2}$
    indices$[m + 1] = \arg\max(\boldsymbol{d})$
    $m = m + 1$

Output: $\boldsymbol{X}$[indices], indices

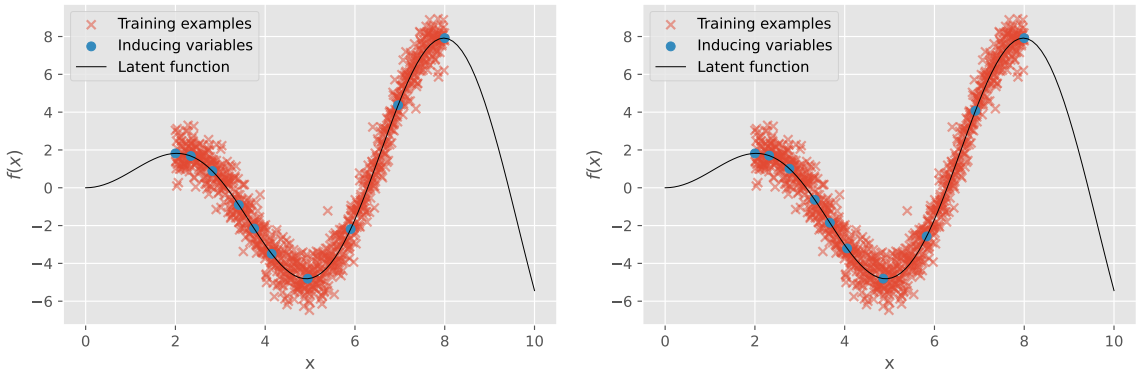
hyperparameters optimisation should be made iteratively:

1. Initialise the inducing points using the greedy variance method
2. Optimise the model hyperparameters
3. Repeat step 1

However, it is inefficient for large-scale selection with NNGP kernels because the kernel computation is usually slow and it needs to select the same amount of inducing points twice. Therefore, we conduct small changes to the algorithm to generate inducing points for the particular case of NNGP kernels. As shown in Figure 3.2, the inducing points selected from two different NNGP kernels are identical since the parameters of the kernel, σ_w and σ_b, only monotonically transform the covariance and do not affect the index of the point that has the maximum variance (Equation 2.12). Therefore, it is reasonable to use a non-optimal kernel to increment inducing points for better efficiency greedily.

3.3 Data Invariance

At an abstract level, a group is defined as G that any operation on two elements of G (denoted as "$\cdot$") produces a third element of G. A group satisfies three axioms: 1) identity element: there exists a unique element $e \in G$ such that $e \cdot a = a$ and $a \cdot e = a$, for every $a \in G$, 2) associativity: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ and 3) inverse element: $a \cdot a^{-1} = e$ and $a^{-1} \cdot a = e$ where $\forall a \in G$, $\exists a^{-1} \in G$ and e is the identity element. In practical applications, each element in the group G can be seen as a transformation



(a) (b)

Figure 3.2: The greedily selected inducing points from two different NNGP kernels. (a) The standard deviations of weights and bias are 1. (b) The standard deviations of weights and bias are 10.

or group action $t : \mathcal{X} \to \mathcal{X}$ acting on a input space $\mathcal{X}$. The outputs from such actions form the orbit $O_x = \{t(\boldsymbol{x}) \mid t \in G\}$ of an element $\boldsymbol{x} \in \mathcal{X}$.

Data augmentation increases the number of training samples by modifying original training inputs in a way that does not change the outputs. Regardless of the data format, any change applied to the data can be treated as a transformation. For example, G is an infinite group that contains all rotations in a fixed 2D plane. The orbit of an image under group G is an infinite set of all rotated versions of the image. If two images can be transformed from each other by an action $t \in G$, the data points share an orbit, and an invariant model is expected to make the same prediction for these images.

Following this intuition, van der Wilk et al. (2) propose an invariant function $f(\cdot)$ from a non-invariant function $g(\cdot)$ by summing over the orbit of a point:

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{x_a} \in O_{\boldsymbol{x}}} g(\boldsymbol{x_a}) \tag{3.14}$$

By placing a GP prior on $g(\cdot) \sim \mathcal{N}(0, k_g(\cdot, \cdot'))$, the invariant function $f(\cdot)$ is also a GP variable since Gaussians are closed under summation

$$k_f(\boldsymbol{x}, \boldsymbol{x'}) = \mathbb{E}\left[\sum_{\boldsymbol{x_a} \in O_{\boldsymbol{x}}} g(\boldsymbol{x_a}) \sum_{\boldsymbol{x'_a} \in O_{\boldsymbol{x'}}} g(\boldsymbol{x'_a})\right] = \sum_{\boldsymbol{x_a} \in O_{\boldsymbol{x}}} \sum_{\boldsymbol{x'_a} \in O_{\boldsymbol{x'}}} k_g(\boldsymbol{x_a}, \boldsymbol{x'_a}) \tag{3.15}$$

However, when the summation exceeds a large orbit size, the kernel function faces a scalability problem. If the orbit is infinite, Equation 3.15 becomes an analytically intractable double integrals over the augmentation distribution $p(\boldsymbol{x_a}|\boldsymbol{x})$. van der Wilk et al. (2) address the intractability by sparse variational Gaussian process and stochastic optimisation (37).

In this project, we stick with the sparse Gaussian process regression. To evaluate the variational lower bound and upper bound, the computations of $\boldsymbol{K_{uu}}$, $\boldsymbol{K_{uf}}$ and

the diagonal of $\boldsymbol{K_{ff}}$ have to be approximated. If we consider the input space of inducing points is different from the source space of training data, the posterior $q(\boldsymbol{f})$ (Equation 3.3) can be constructed in $g(\cdot)$ instead of $f(\cdot)$ by inter-domain inducing points (52)

$$\begin{aligned} k_{uf}(\boldsymbol{z}, \boldsymbol{x}) &= \mathbb{E}_{p(g)}[g(\boldsymbol{z})f(\boldsymbol{x})] = \sum_{\boldsymbol{x_a} \in O_{\boldsymbol{x}}} k_g(\boldsymbol{z}, \boldsymbol{x_a}) \\ k_{uu}(\boldsymbol{z}, \boldsymbol{z'}) &= \mathbb{E}_{p(g)}[g(\boldsymbol{z})g(\boldsymbol{z'})] = k_g(\boldsymbol{z}, \boldsymbol{z'}) \end{aligned} \tag{3.16}$$

The new $\boldsymbol{K_{uf}}$ and $\boldsymbol{K_{uu}}$ only require a single summation and no summation, respectively. But, the problem of intractable computation still exists if the orbit size of training data is too big or infinite. van der Wilk et al. (2) use simple Monte Carlo estimate to construct unbiased estimations for intractable terms

$$\begin{aligned} \hat{k}_{uf}(\boldsymbol{z}, \boldsymbol{x}) &= \sum_{s=1}^{S} k_g(\boldsymbol{z}, \boldsymbol{x^{(s)}}) \\ \hat{k}_{ff}(\boldsymbol{x}, \boldsymbol{x}) &= \frac{1}{S(S-1)} \sum_{s=1}^{S} \sum_{s'=1}^{S} k_g(\boldsymbol{x^{(s)}}, \boldsymbol{x^{(s')}})(1 - \delta_{ss'}) \end{aligned} \tag{3.17}$$

where $\boldsymbol{x^{(s)}} \sim p(\boldsymbol{x_a}|\boldsymbol{x})$. Therefore, we obtain the estimations of all intractable terms to compute the variational lower bound and upper bound. For simplicity, we will refer to a SNNGP with this summation kernel as an invariant sparse neural network equivalent Gaussian process (iSNNGP).

Chapter 4

Evaluations

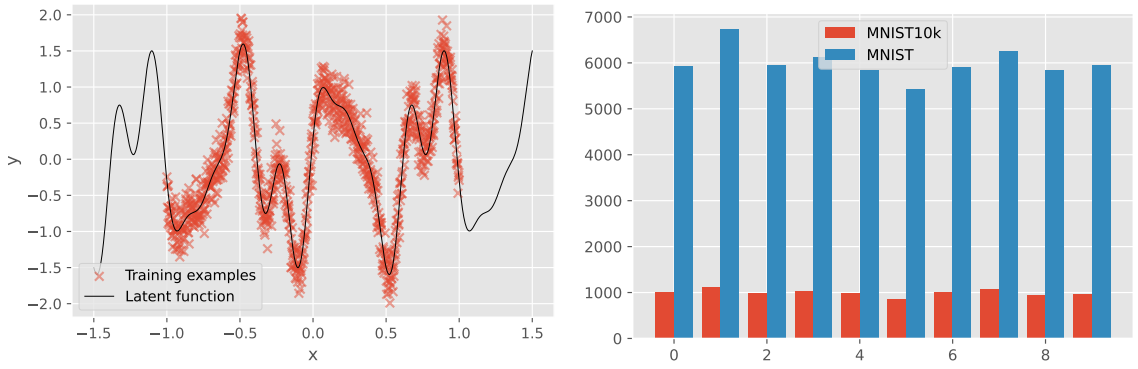
4.1 Experiment Setup

In this project, we study regression and classification problems on Gaussian processes with NNGP kernels and summation kernels. The regression tasks try to find the latent function that maps the input values to the continuous output variables. In contrast, the classification tasks look for the decision boundaries that separate the dataset into different discrete classes. Since the consumer-level graphic cards only have up to 24 Gigabytes of memory and our experiments will consume much more memory, we conduct the experiments on a single CPU with 8 logical cores and 64 gigabytes of RAM.

4.1.1 Datasets

We implement a one-dimensional synthetic dataset as the regression dataset. The latent function is defined as $f(x) = \sin(3\pi x) + 0.3\cos(9\pi x) + 0.5\sin(7\pi x)$. The 1000 training data are sampled uniformly between $x = -1$ and $x = 1$. Gaussian noises are added to the function values to construct the training targets, $y = f(x) + \epsilon, \epsilon \sim \mathcal{N}(0, 0.04)$. For the test dataset, we sample 1000 points uniformly between $x = -1.5$ and $x = 1.5$ and the test targets are simply noise-less function values. Therefore, we can testify to the performance of models for interpolations and extrapolations. The illustration of the 1D dataset is shown in Figure 4.1(a).

For the classification task, we use the MNIST dataset (53) with 60000 training data and 10000 test data for handwritten digit recognition. Each image is in a 28×28 pixel format. We divide the pixel values by 255 to normalise the images because they have different grey scales. Despite changing the discrete target values to one-hot encoding, each multivariate target is also subtracted by 0.1 (i.e. 0.9 for positive data and -0.1 for negative data) to ensure the outputs have zero mean. Due to limit computational resources, we also conduct experiments on the first 10000 training data (referred to as MNIST10k) for better interpretation of performance. Since the MNIST data are shuffled already, retrieving a subset of the training data does not



(a) (b)

Figure 4.1: Data statistics. (a) The plot of 1D synthetic dataset. (b) The class distributions of MNIST and MNIST10k.

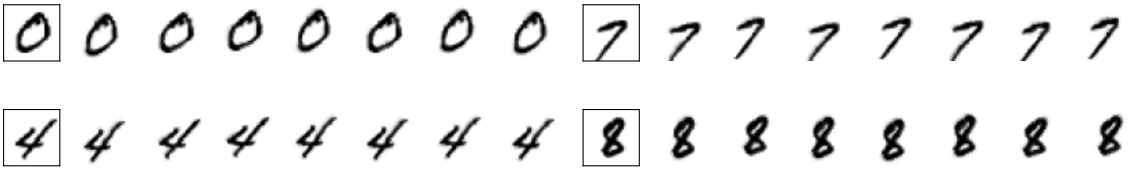


Figure 4.2: Examples of transformed MNIST digits from the infinite MNIST dataset. The digits in the squares are the original data.

break the balance of labels (Figure 4.1(b)).

The infinite MNIST dataset is used as the augmentation dataset as it can provide an infinite number of pseudo-random deformations and translations of the MNIST digits. A long integer index identifies each sample. The indices from 0 to 9999 and from 10000 to 69999 are the standard MNIST test data and training data, respectively. If a sample has a index $i \geq 70000$, the sample is generated by pseudo-random transformations to the MNIST training sample $10000 + ((i - 10000)\%60000)$. Figure 4.2 shows a few examples of the MNIST digits after transformation. Even though the digits in each group are visually similar, there are pixel-level differences between each pair of images.

4.1.2 Neural Networks

Neural Tangents (40) is a high-level neural network API that allows researchers to easily specify multi-layer infinitely width neural networks and finite ones. Given the datasets we used in the project, we use this package to implement feedforward neural networks and convolutional neural networks.

Infinitely wide neural networks characterise some different designs in the Neural Tangents. First, neural network modules are initialised using Lecun/He initialisation (54, 55), which is different from the standard JAX library that uses Xavier ini-

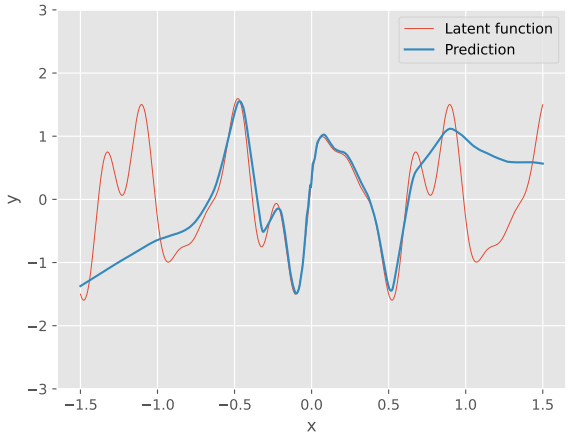


Figure 4.3: The predictions of the finitely wide FFNN on the synthetic data.

Dataset	FFNN	CNN
MNIST10k	95.40%	96.96%
MNSIT	98.05%	98.37%

Table 4.1: The accuracies of FFNNs and CNNs on the MNIST10k and MNIST datasets.

tialisation (56). Also, the Neural Tangent Kernel parameterisation (57, 58) is used by default in the randomly weighted layers instead of the standard parameterisation in the JAX. So, different learning rates should be tried to get similar results for training finitely wide neural networks in the Neural Tangents.

In order to find appropriate hyperparameters of the infinitely wide networks, we first train their finite counterparts on these datasets and pick the best model based on the performance of the validation set (10% of training data). We implement a 20-layer feedforward network for the synthetic dataset with 512 hidden units at each layer and use ReLU as activations. The data are split into batches with a size of 100. The network is optimised by stochastic gradient descent with 0.1 learning rate and 0.9 momentum for 100 epochs. Figure 4.3 shows that the network is able to fit the range of x between -0.5 to 0.5. However, it struggles to fit the training data at two edges. The rooted mean squared error (RMSE) of this network on the test data is 0.79.

For the MNSIT10k and MNIST datasets, we consider both FFNN and CNN. The FFNN for this task has 3 linear layers, and each hidden layer has 1024 hidden units. The architecture of the CNN is shown in Figure 4.4. We choose the ReLU function as the nonlinear activation between layers for both networks. In this case, we set the batch size to 128, and SGD optimises networks with a 1 learning rate and 0.9 momentum

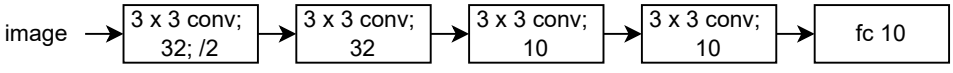


Figure 4.4: The architecture of the CNN for the MNIST dataset. The details of each layer is specified as $\langle filter\_size \rangle$ *conv*; $\langle out\_channels \rangle$; $\langle strides \rangle$.

Num. of layers	Opt. (σ_w, σ_b)	LML	ELBO
3	$(4.53, 6.77)$	97.0791	97.0696
11	$(1.81, 1.75)$	100.3334	100.3239

Table 4.2: The optimised kernel parameters and approximation metrics for 3-layer SNNGP and 11-layer SNNGP. Inducing points are the entire training data and LML stands for log marginal likelihood.

for 100 epochs. Table 4.1 demonstrates the test accuracy of the models on two datasets. While both models achieve over 98% accuracy on the MNIST dataset, the CNN is slightly better than the FFNN in generalisation.

Since a multi-layer feedforward neural network is a universal function approximation, and we only consider the case when the number of hidden units or channels tends to infinity, it is reasonable to reduce the depth of networks. So, we remove the last two convolutional layers for the CNN with infinite filters and use a 3-layer infinitely wide FFNN. While the two models are theoretically similar, the inference and optimisation on the GP analogue of convolutional have shown difficulty on the dataset that its size is larger than 2000 due to computational inefficiency (25). Therefore, we will only focus on the study of infinitely wide FFNN and expect similar results for convolutional cases.

4.2 Characteristics of Hyperparameters

We study three hyperparameters of the NNGP kernel (i.e. σ_w, σ_b and the number of layers) to understand their characteristics better. In this case, we conduct experiments on the 1-dimensional synthetic dataset for visual interpretation, and all experiments are based on sparse approximation with the full training data as inducing points.

First, we tune these hyperparameters individually to observe the data fitting and uncertainty estimates, as shown in Figure 4.5. When the number of layers is fixed at 3 and the value of σ_w raises, we observe that the predictions become more flexible and the uncertainties also significantly increase. When σ_w is 1, the posterior underfit the training samples with high confidence. When σ_w is 10, the predictions become too flexible and fit the noises in the training data. At extrapolation, the uncertainty estimates increase as the value of σ_w rises.

Similar overfitting is also noticed when the number of layers increases to 6 because deeper neural networks are naturally complex. Enlarging σ_b does not significantly affect the curve fitting and uncertainty estimates. However, the predictive distribution with a smaller σ_b is overconfident at extrapolations (Figure 4.5(e)).

Table 4.2 demonstrates the performance of two SNNGPs with different numbers of layers and optimised kernel parameters. Both models can approximate the posterior with 0.0095 KL-divergence. From the results of three optimised hyperparameters, the shallow model has higher σ_w to make the posterior complex enough to fit the

data. Since the deep model is already flexible, the σ_w and σ_b are much smaller than those of the shallow model. So, we can trade off the number of layers and standard deviations to reduce the recursion of kernel computation for better efficiency.

4.3 Regression of Synthetic Dataset

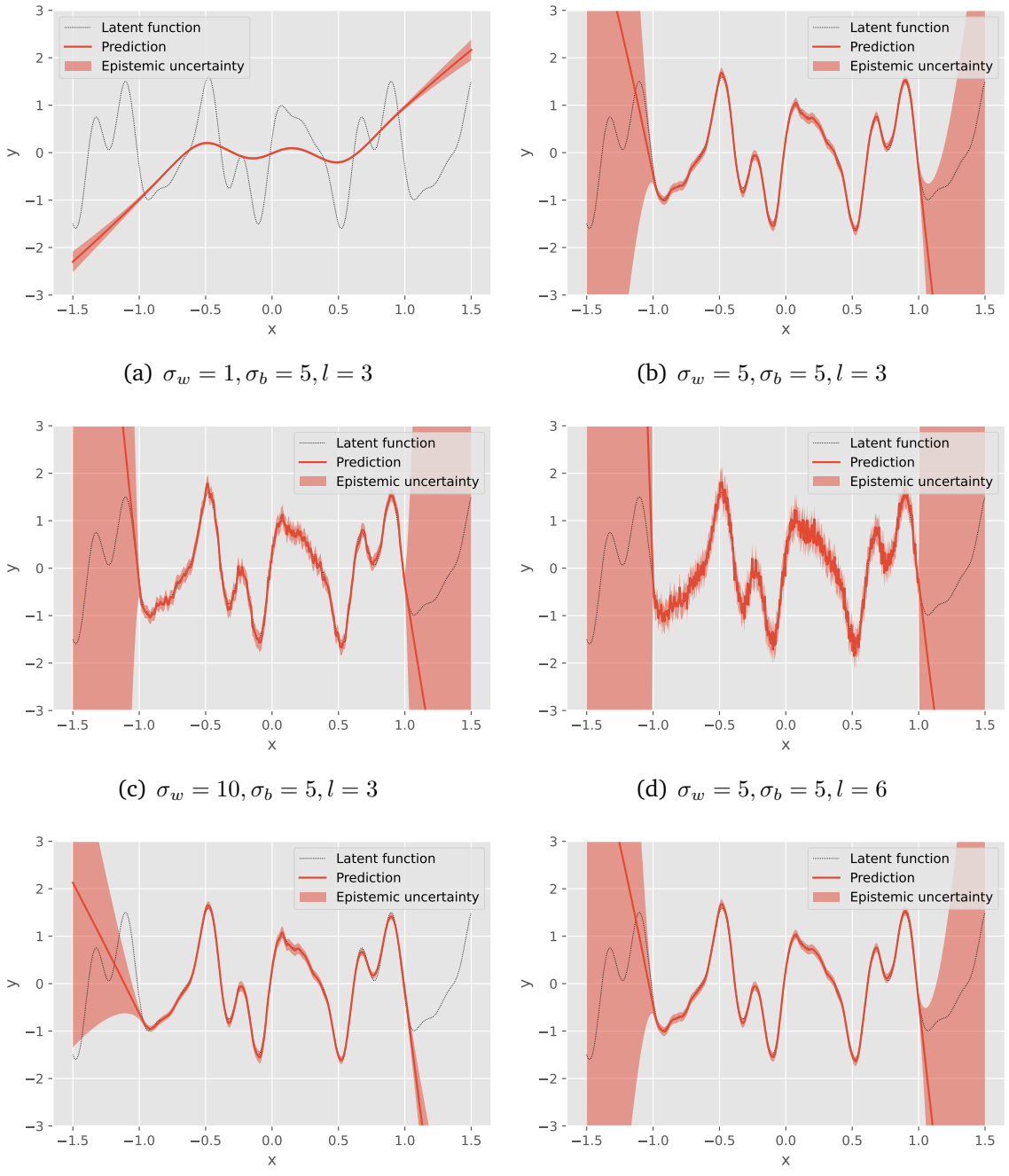
In this section, we analyse the performance of SNNGP on the synthetic dataset as the number of inducing points grows (Figure 4.6). For every experiment, the number of inducing points increments from 10 to 1000 and is numbered evenly on a $\log 10$ scale of the training data size. The standard deviations of weights and bias and the likelihood variance are optimised at each point.

As the number of inducing points rises to 100, the model fits the training data progressively better since its log marginal likelihood increases from -600 to 97 approximately. Because the complexity of the predictive curve is highly related to the values of σ_w when the number of layers is fixed, we notice that the model underfits the training data when the number of inducing points is small (Figure 4.6(f)). As more inducing points will cover more regions of the input space, the model realises the latent function is more sophisticated. After 100 inducing points, the complexity of the latent function can be determined, and σ_w will not keep growing because of the complexity penalty in the ELBO. On the other hand, the estimated likelihood variance decreases from around 0.24 to 0.038 as M increases. So, the SNNGP can roughly estimate the true likelihood variance if sufficient inducing points are provided.

The tendencies of ELBO and EUBO depend on the log marginal likelihoods. We observe that the ELBO and EUBO grow as the log marginal likelihood dramatically increases (Figure 4.6(a) & 4.6(b)). Before the log marginal likelihood starts to converge, the gap between ELBO and EUBO does not reduce and remains above 500 since a much better data fitting can be found as the number of inducing points increases (Figure 4.6(c)). While more inducing points cannot provide useful information in their regions after $M = 100$, the log marginal likelihood converges to 97.08.

The ELBO and EUBO also start converging to the log marginal likelihood as the number of inducing points exceeds 100. As shown in Figure 4.6(d), the KL divergence between the approximate and the true posterior reduces significantly from around 200 to 0.01 when the number of inducing points increments from 50. Figure 4.6(b) demonstrates that EUBO is farther away from the log marginal likelihood than the distance between the log marginal likelihood and ELBO. Our results also testify to Titsias (36) that the EUBO converges to the marginal likelihood slower than the ELBO with an increasing number of inducing points since the gap between EUBO and ELBO is only close to 0 with 1000 inducing points (Figure 4.6(c)).

As more inducing points, rooted mean squared errors (RMSE) increase from 0.78 to 3.28. If we compare Figure 4.5(a) with Figure 4.5(b), the predictions from the model with small σ_w fluctuates around zero and the predictions from the model with large σ_w tends to extreme values at extrapolation. The same scenario is also applied here.



(a) $\sigma_w = 1, \sigma_b = 5, l = 3$

(b) $\sigma_w = 5, \sigma_b = 5, l = 3$

(c) $\sigma_w = 10, \sigma_b = 5, l = 3$

(d) $\sigma_w = 5, \sigma_b = 5, l = 6$

(e) $\sigma_w = 5, \sigma_b = 1, l = 3$

(f) $\sigma_w = 5, \sigma_b = 10, l = 3$

Figure 4.5: The approximated predictions of SNNGP with different kernel settings. Inducing points are the entire training data and l is the number of layers.

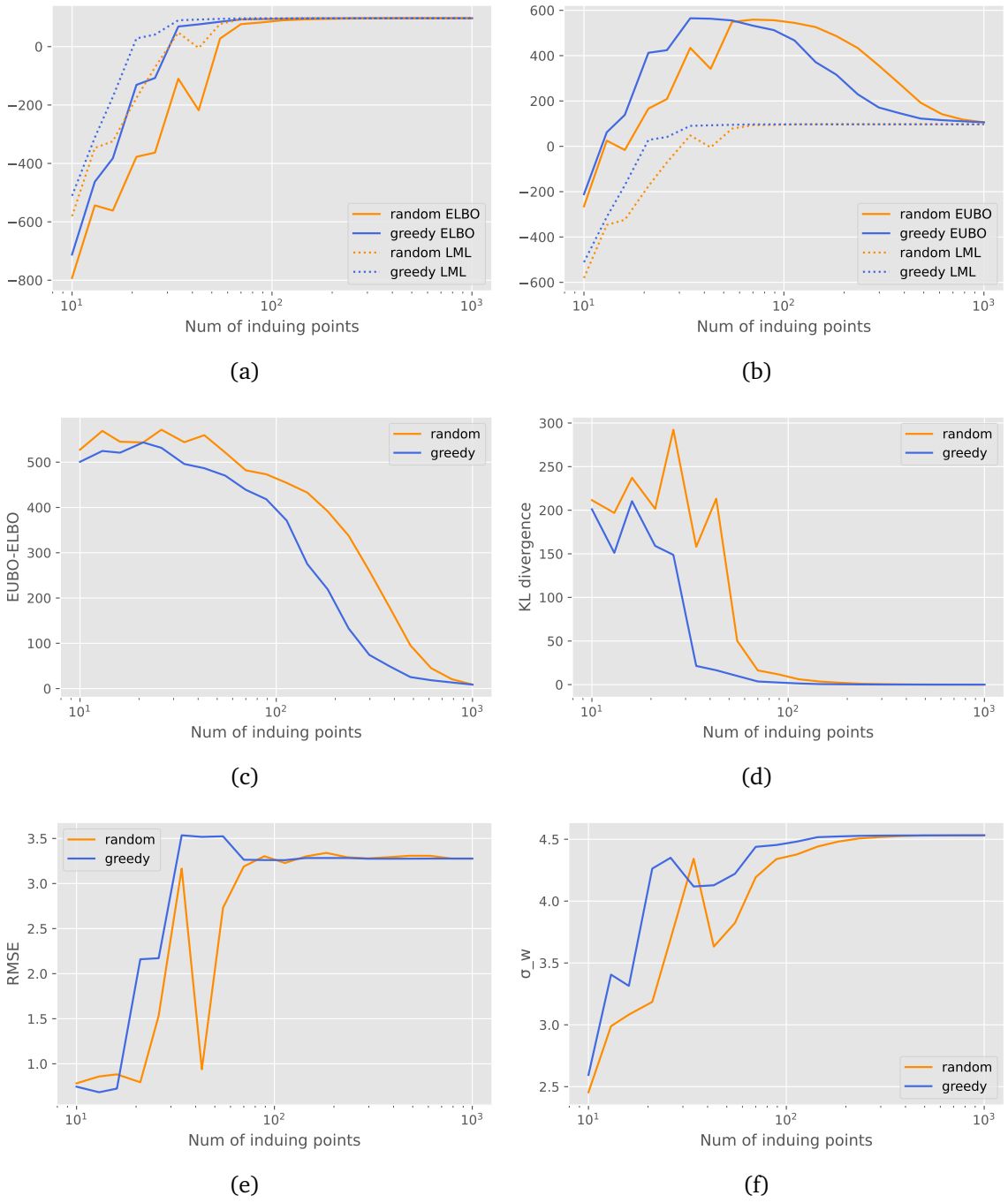


Figure 4.6: The performance of 3-layer infinitely wide neural networks on the synthetic dataset. Random indicates random inducing points selection. Greedy indicates greedy inducing points selection. LML is the log marginal likelihood computed on the full training data.

When the number of inducing points is small, the model underfits the training data. However, the average distance between the predictions and true values is also small for this special case. The poor performance of the complex model at extrapolations significantly increases the RMSE loss even if it fits the training data well.

The performance fluctuation with randomly selected inducing points is noticed at around 50 inducing points. Random selection has some uncertainty on the diversity of inducing points. Inducing points may be clustered in a region or miss the extreme points of the latent function. So, the network may lack information in certain regions. However, the SNNGP with greedy selection has a more stable convergence and test accuracy than random selection due to diverse and related inducing points, which cover the extremes and saddle points. So, the approximate with greedy selection is consistently better than that with random selection.

4.4 Classification of MNIST10k Digits

In this section, we evaluate the approximation and test performance of SNNGPs and iSNNGPs on MNIST10k and the different sizes of augmentations. Since the targets of MNIST digits are multivariate, we use 10 conditionally independent Gaussian processes to estimate 10 classes. In this case, we increment the number of inducing points to 10000 with a fixed step size of 5% of the training data size.

4.4.1 Sparse Approximation

We observe a completely different situation on the MNIST10k than the approximation on the synthetic dataset (Figure 4.7). From the tendency of σ_w (Figure 4.7(f)), the network becomes more complex as increasing number of inducing points. Since the log marginal likelihood keeps increasing, the ELBO and EUBO also consistently rise. However, the KL divergence decreases continuously from over 1000 to near 0, representing that the ELBO is asymptotically close to the log marginal likelihood even if they grow simultaneously. The gap between the ELBO and EUBO remains above 5000 before the exact GP posterior is well approximated. When the number of inducing points is equal to 10000, the KL divergence significantly decreases to 389.47 and the interval between EUBO and ELBO reduces from 52751 to 23390.

From the results on the synthetic dataset, we know that the SNNGP with greedy selection is robust when M is small. However, it is not always the case on the MNIST10k. When M is minor than 2000, the data fitting, approximation and test accuracy of the random selection are better than those of greedy variance. While the number of inducing points reaches 10000, greedy selection has more stable improvements and surpasses the random selection after 5000 inducing points. Figure 4.8 shows that greedy selection does not have significant class preferences. So, the variances of training data in each class should be similar.

Since the SNNGP cannot be optimised with 10000 inducing points on the MNIST dataset due to extensive memory consumption, observations on the MNIST10k can

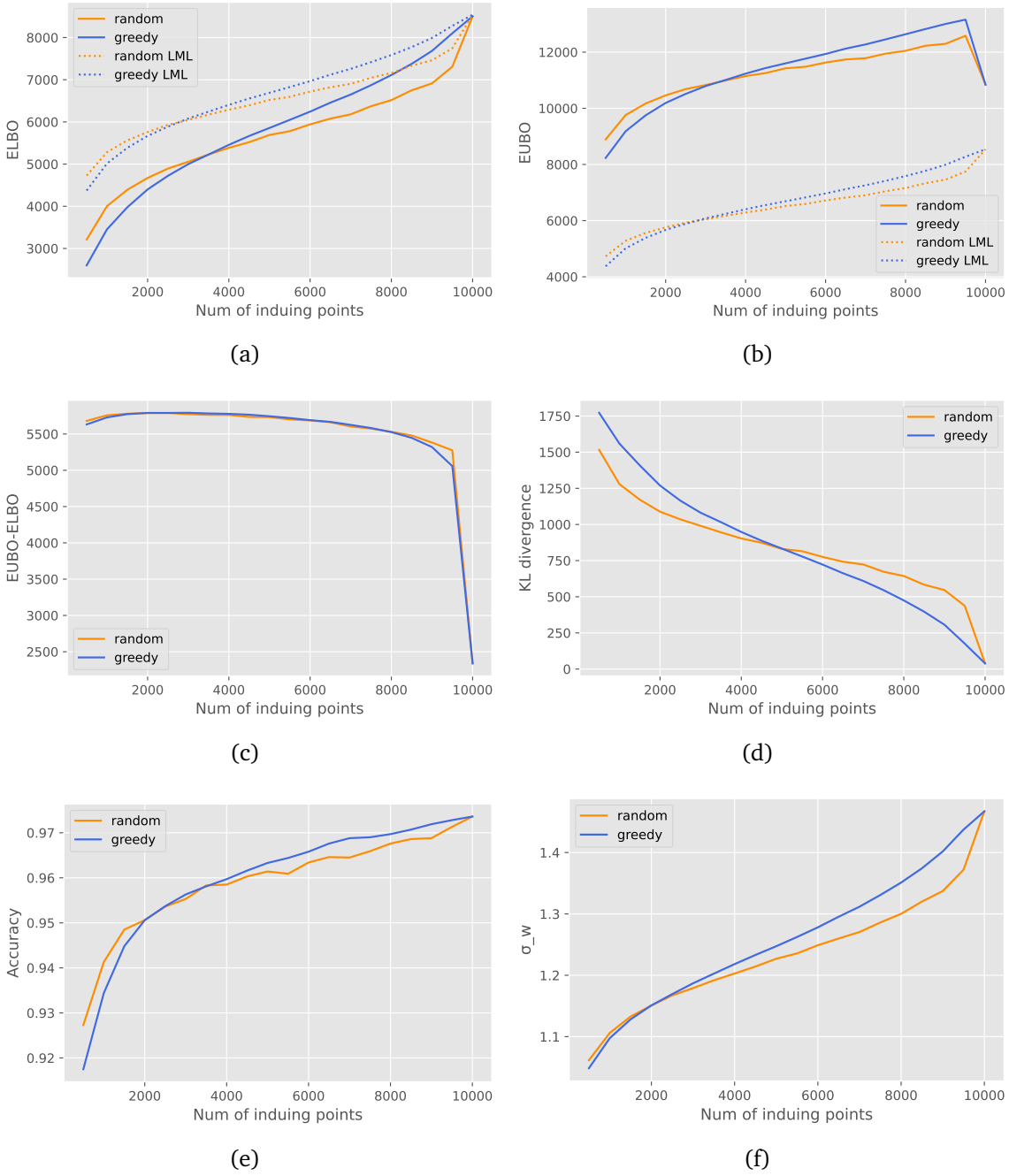
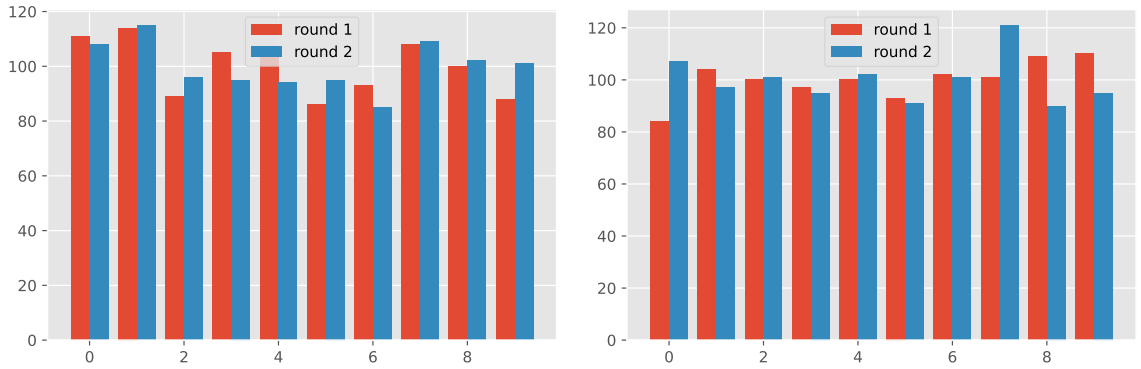


Figure 4.7: The performance of 3-layer infinitely wide neural networks on MNIST10k. Random indicates random inducing points selection. Greedy indicates greedy inducing points selection. LML is the log marginal likelihood computed with the full training data.



(a) MNIST10k (b) MNIST

Figure 4.8: The class distribution of 1000 inducing points from two independent greedy selections on MNIST10k and MNIST, respectively.

be induced to the MNIST. As shown in Figure 4.1(b), the MNIST dataset is well-balanced and shuffled. Although the choices of inducing points on MNIST10k are not as diverse as the selection on the MNIST, randomly selecting inducing points from a random subset is still a random selection. For the greedy selection, it still prefers the inducing points with large variances, which essentially is close to the greedy inducing points from the entire dataset. Since the approximation of SNNGPs depends on the quality of inducing points and both strategies provide similar inducing points on the MNIST10k and MNIST, the inducing points from the subset still contain similar information as those from the full dataset that can describe their respective regions. If the GP posterior cannot be approximated within 10000 inducing points on the MNIST10k, we expect that it also cannot be approximated within 10000 inducing points on the MNIST.

4.4.2 Data Augmentations

In this section, we evaluate the performance of SNNGPs on the MNIST10k with an additional 10000 augmented data (AUG10k) and randomly selected inducing points. By comparing its approximation with that on MNIST10k, we observe a nearly similar tendency as the number of inducing points raises (Figure 4.9(a) & 4.9(b)). Before M is equal to the data size, the gap between EUBO and ELBO only slightly changes since a better fitting is always discovered as more inducing points. However, the approximate slowly converges to the true posterior distribution since the KL divergence decreases. When the number of inducing points is equal to the data size, the KL divergence decreases to 877 on AUG10k. While the gap between boundaries significantly reduces, the interval is still around 60000.

Figure 4.9(c) demonstrates the accuracy of the MNIST test dataset of the model trained on MNIST10k and AUG10k, respectively. The model on the AUG10k is marginally better than its baseline at each step. The final result shows that the model on AUG10k achieves 97.84% test accuracy while the network on MNIST10k

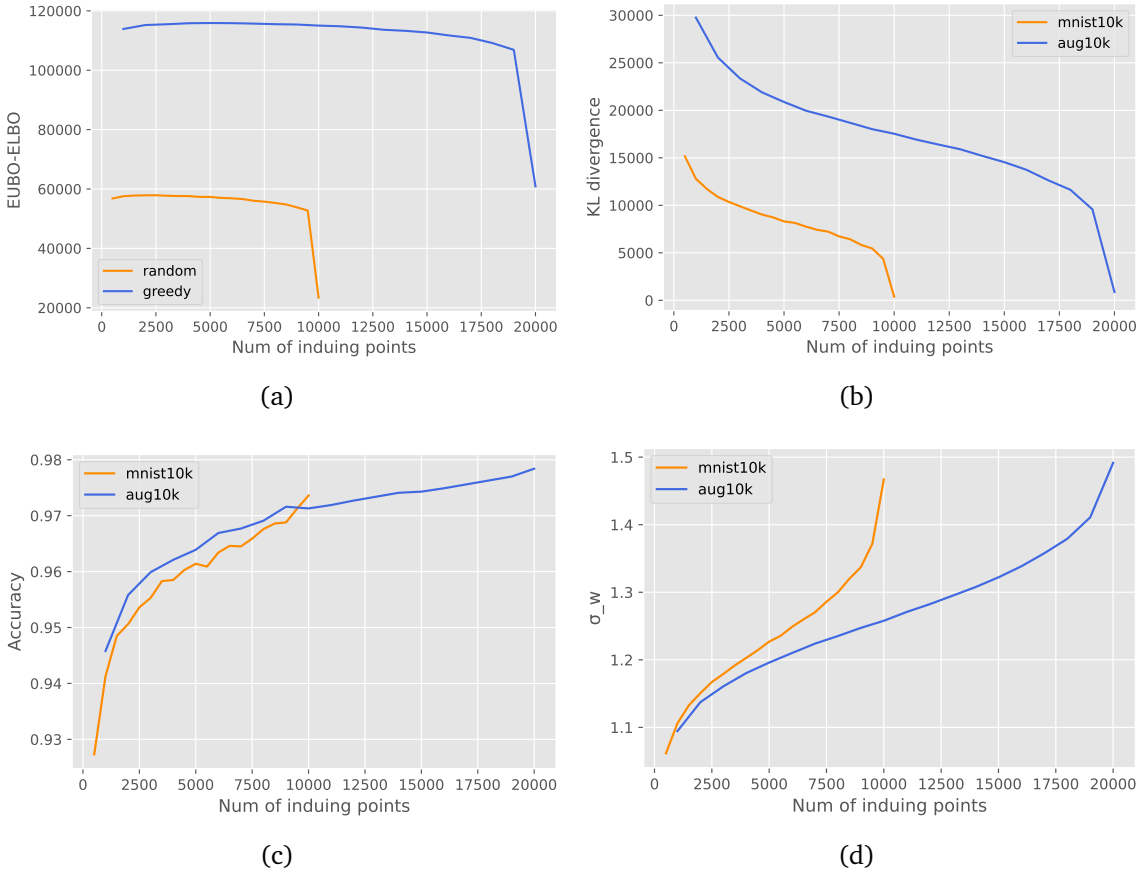


Figure 4.9: The performance of 3-layer infinitely wide neural networks on the MNIST10k and MNIST10k with 10000 augmentations (AUG10k).

has 97.36% accuracy. Figure 4.9(d) indicates that the model on AUG10k requires a much larger amount of inducing points to gain the similar σ_w as the model on the MNIST10k. However, the model on AUG10k does not underfit training data due to its adequate test performance, even though the σ_w is small. Although the optimal σ_w increases slower on the larger dataset, we speculate that the larger covariance matrix between the inducing points and training data can compensate for the data fitting.

A natural question arises if our observation on AUG10k applies to the approximation on the MNIST with 20000 inducing points. Since the MNIST dataset is well designed and has been used for many years, our augmented data is more likely to be noisy. Therefore, if our deformed images always contain helpful information and prevent us from approximating the true posterior with $M < N$, we do not expect the approximation will converge within 20000 inducing points on the full MNIST dataset.

4.4.3 Learning Invariances

In this section, we compare the performance of the summation kernel on different sizes of data augmentations with the performance of the standard NNGP kernel

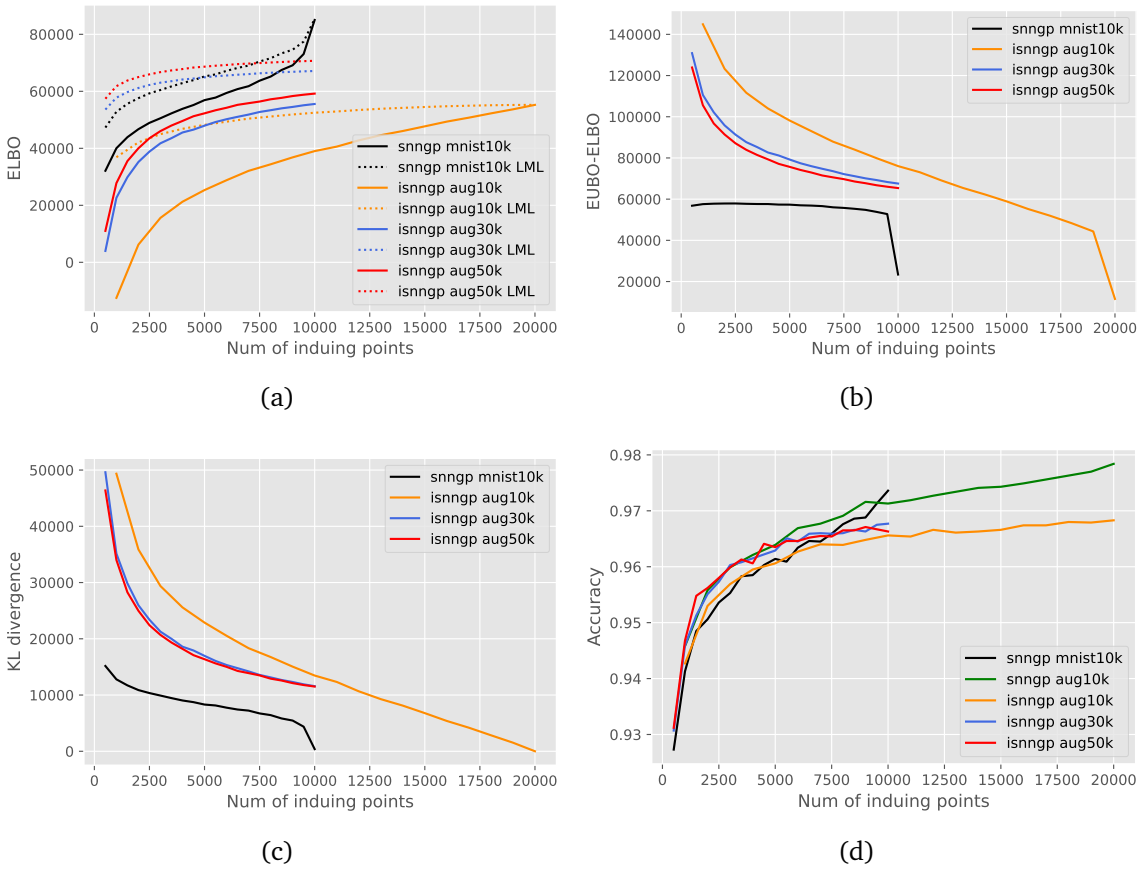


Figure 4.10: The performance of 3-layer infinitely wide neural networks on the MNIST10k with standard NNGP kernel and AUG10k, AUG30k and AUG50k with the summation kernel, respectively. The test performance with standard augmentation is also compared in (d). LML is the log marginal likelihood computed with the full training data.

on the MNIST10k. Like the setting in the previous section, we first use the same 10000 deformed digits as augmentations and choose the inducing points from training and augmented data. From the curve of log marginal likelihood (dotted line in Figure 4.10(a)), we notice that the log marginal likelihood on the AUG10k is much smaller than that on the MNIST10k and is gradually flattened as the number of inducing points grows.

The flattened log marginal likelihood primarily affects the tendencies of the EUBO, which is now asymptotically close to the log marginal likelihood. The interval between boundaries converges logarithmically when M is small, in contrast to the unchanging interval of the SNNGP (Figure 4.10(b)). Figure 4.10(c) shows that the approximation of the iSNNGP on AUG10k converges asymptotically to the true posterior. But, its KL divergence is much higher than that of the SNNGP on the MNIST10k with the same M. As we further push the number of inducing points beyond 10000, the rapid change in the approximation of the iSNNGP is not observed and a good approximation is only obtained at 20000 inducing points. The test accuracy of the

iSNNGP 4.10(d) is also gradually saturated around 96.8% when the number of inducing points exceeds 10000. So, the improvement of learned invariances is not shown on the AUG10k.

The poor performance of the iSNNGP on AUG10k may be caused by the size of training data is not scaled enough. Since the sizes of kernel matrices remain unchanged no matter how many augmentations are added, we scale the size of augmentations to 30000 (AUG30k) and 50000 (AUG50k), respectively, for the following experiments. Due to inter-domain inducing points, the size of $\boldsymbol{K_{uu}}$ is larger than the size of $\boldsymbol{K_{fu}}$ if we set the number of inducing points to be greater than the size of training data. It violates the principle of sparse approximation. Therefore, we will limit the number of inducing points to 10000 for the following experiments.

As we increase the number of augmentations, we notice that the log marginal likelihoods on AUG30k and AUG50k significantly improve by more than 10000, compared with that on the AUG10k (Figure 4.10(a)). Before $M = 5000$, their log marginal likelihoods are higher than that of the SNNGP without augmentation. Then, the log marginal likelihoods of iSNNGPs tend to saturation again while that of the SNNGP is still increasing. As shown in Figure 4.10(b) and 4.10(c), the approximates also asymptotically converge the posteriors. The convergence speed of an iSNNGP is slower on larger datasets as it needs more inducing points.

From the perspective of test accuracy, iSNNGPs on AUG30k and AUG50k achieve competitive performance compared with SNNGP on AUG10k before 5000 inducing points. Between 5000 inducing points and 7500 inducing points, the iSNNGPs are still better than the SNNGP on MNIST10k. After that, the iSNNGP is challenging to make further improvements while the test accuracy of SNNGPs with or without data augmentation consistently increases and exceeds the accuracies of iSNNGPs. Although the data fitting is slightly better by increasing the augmentation set from 30000 to 50000, the test performance does not gain clear benefits.

4.5 Classification of MNIST Digits

In this section, we further scale training data size and evaluate the approximation and generalisation of infinitely wide neural networks on the full MNIST dataset. Due to limited computational resources, the calculation of marginal likelihood is infeasible. So, only ELBO and EUBO are calculated for evaluating the approximates. We optimise the hyperparameters until 7000 inducing points and use the optimised hyperparameters at that point for the following calculations. The evaluation stops at 20000 inducing points. By conducting experiments on the full MNIST dataset, we can verify our expectations made in the previous sections and observe whether the results will differ.

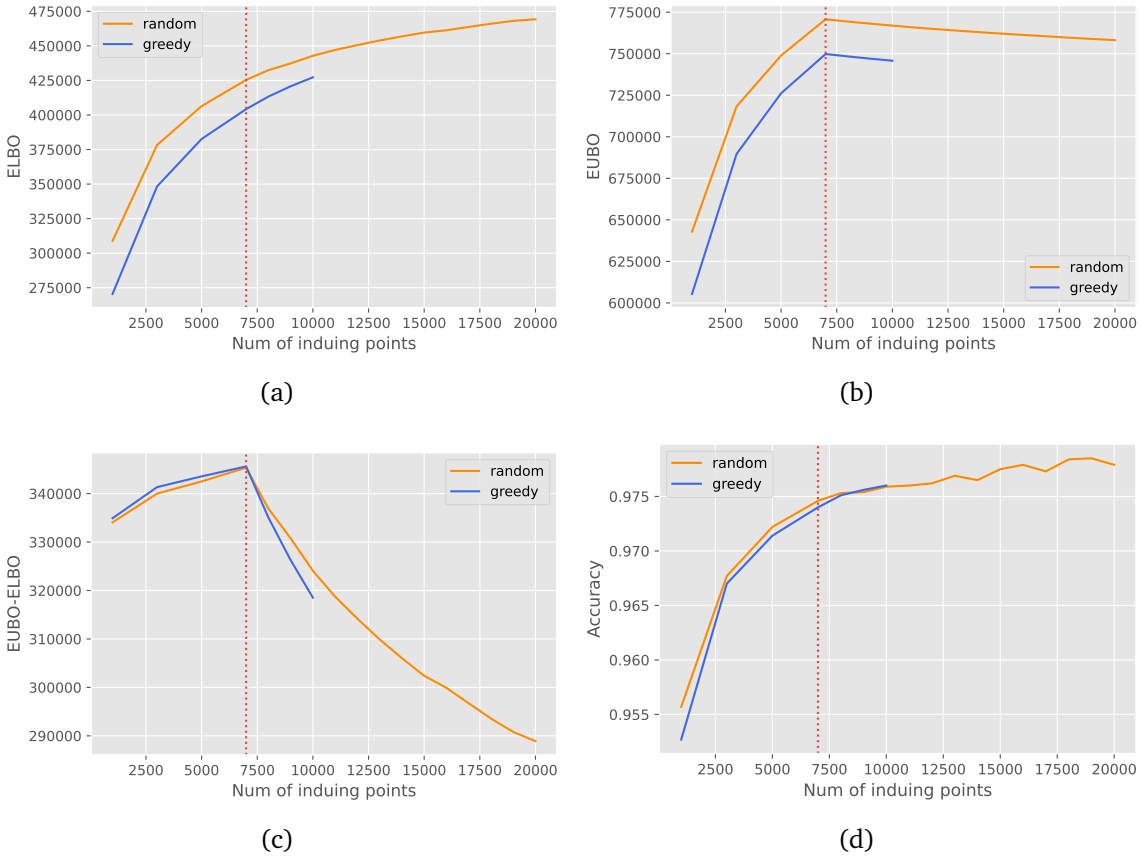


Figure 4.11: The performance of 3-layer infinitely wide neural networks on the MNIST with standard NNGP kernel. Random indicates random inducing points selection. Greedy indicates greedy inducing points selection. Red line indicates the location that optimisation stops.

4.5.1 Sparse Approximation

We compare the performance of two inducing points selections on the MNIST dataset (Figure 4.11). For greedy selection, we only conduct experiments up to 10000 inducing points because the inducing points are selected iteratively and selecting 10000 inducing points from the MNIST dataset takes roughly 9 hours on our experimental machine. We believe the major drawback of greedy selection with an NNGP kernel is the expensiveness of kernel computation on large-scale datasets. However, the greedy selection should be beneficial when the number of inducing points is small. Therefore, conducting partial experiments does not jeopardize our analysis.

Figure 4.11(a) and 4.11(b) demonstrate the variational boundaries on the MNIST dataset with two selection methods. While M is reaching 7000, boundaries keep increasing since more inducing points lead to larger log marginal likelihood, like the situation on MNIST10k. Surprisingly, the ELBO and EUBO with random selection are higher than those with greedy selection. Consequently, the marginal likelihood of the SNNGP with random selection is also higher. It indicates that the data fitting of the SNNGP with random selection is better than the SNNGP with the greedy

selection before 7000 inducing points.

When M exceeds 7000, the log marginal likelihood is fixed since we do not optimise the hyperparameters at each step anymore. While the ELBO is still growing, the EUBO is starting to decrease. As shown in Figure 4.11(c), the gap between boundaries decreases from roughly 34500. The approximate gradually moves to the true posterior with this fixed kernel. Compared with random selection, the convergence of the SNNGP with greedy selection is faster than the SNNGP with random selection as more inducing points. When $M = 20000$, the gap between the ELBO and the EUBO is around 290000, which indicates the approximation is still biased from the true posterior and proves our expectation that it is difficult to find a good approximation within 20000 inducing points on the MNIST dataset.

Because the SNNGP with random selection fits the training data better, it also has slightly higher accuracy when $M < 7000$. After the optimisation is terminated, the test accuracies of the two selection methods become similar due to quicker convergence of the SNNGP with greedy selection. Since the complexity of the model is fixed after $M = 7000$, the improvement of test performance is bounded as more inducing points.

4.5.2 Data Augmentations

In this section, we use 60000 augmentations (AUG60k) to evaluate the approximation of SNNGPs with random selection on standard data augmentation. Due to doubled training data and limited computational resources, we only optimise the hyperparameters until 5000 inducing points for AUG60k. Figure 4.12 demonstrates the comparison between the performance of SNNGPs on the MNIST and AUG60k.

The similarity of the results on the MNIST and AUG60k is still observed. The log marginal likelihood increases before $M = 5000$ as more inducing points lead to better fitting. Figure 4.12(c) shows that the interval between boundaries slightly raises when the hyperparameters are optimised before 5000 inducing points on the AUG60k. After the optimisation stops, the approximated posterior moves towards the true posterior since the ELBO and EUBO starts to converge and the gap between them decreases from 67533 to 55037.

The test accuracy of the SNNGP on the AUG60k is constantly better than the performance on the original dataset (Figure 4.12(d)), even though the optimisation is terminated earlier on the augmented dataset. Before $M = 5000$, the test accuracy significantly increases, which matches the log marginal likelihood of the SNNGP on AUG60k. After that, the test accuracy of the SNNGP on AUG60k is still higher than that on MNIST as increasing inducing points. When M is close to 20000, the test accuracy exceeds 98% with the "sub-optimal" hyperparameters.

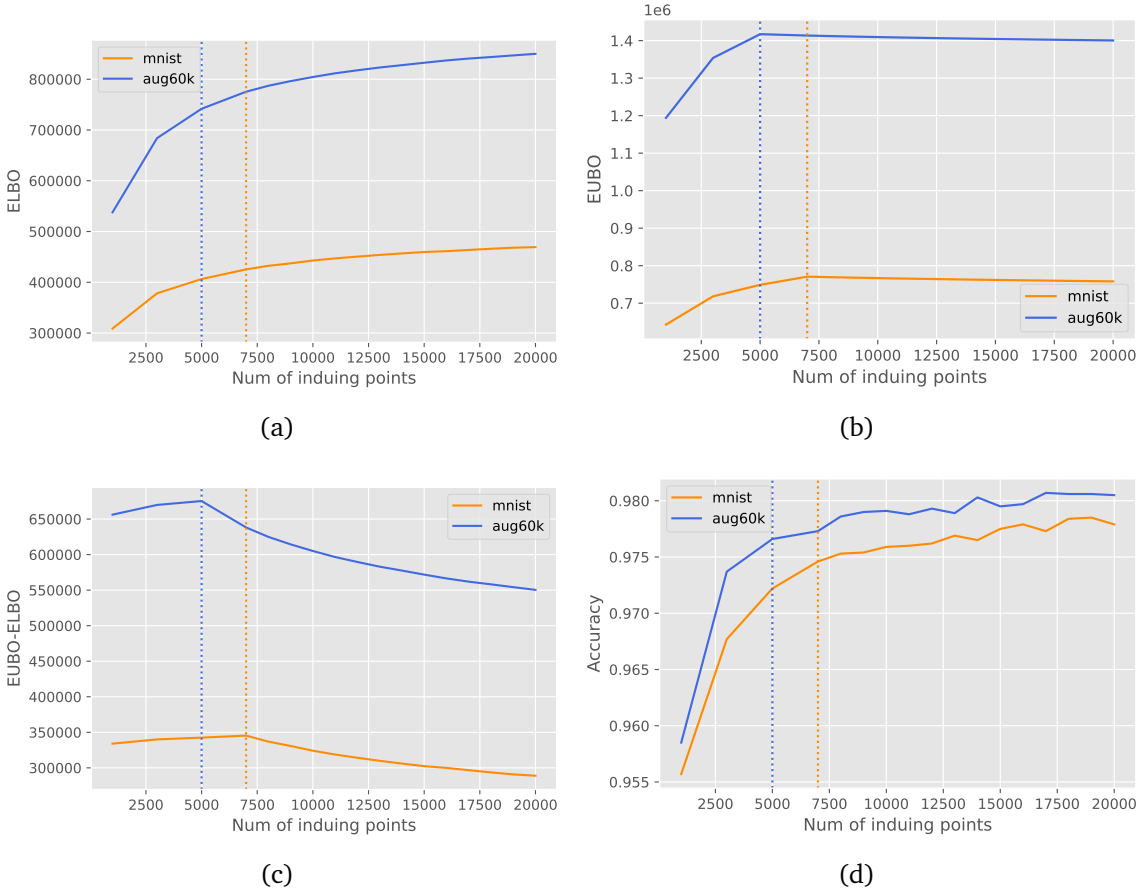


Figure 4.12: The performances of 3-layer infinitely wide neural networks with NNGP kernel and random selection on the MNIST and AUG60k. Vertical lines indicate where the optimisations stop given the datasets, respectively.

4.5.3 Learning Invariances

For the iSNNGP on the augmented MNIST dataset, we triple the training data with 120000 augmentations (AUG120k) because only 60000 augmentations will not improve the performance from the results on MNIST10k and increasing the size of training data does not significantly worsen the computational complexity of the variational boundaries. In this case, we still terminate the optimisation at 7000 inducing points and increment M up to 20000.

Figure 4.13(a) and 4.13(b) compare the variational boundaries of the iSNNGP on AUG120k and the SNNGP on the standard MNIST dataset. It is difficult to determine whether the log marginal likelihood of the iSNNGP is better than the SNNGP since the boundaries of the SNNGP are "sandwiched" by the boundaries of the iSNNGP. However, from the results on the MNIST10k, we can expect that the iSNNGP on the AUG60k obtains a higher log marginal likelihood than the SNNGP on the MNIST.

We also observe that the interval between boundaries of the iSNNGP keeps decreasing as M grows, in contrast to the results of the SNNGP (Figure 4.13(c)). Therefore,

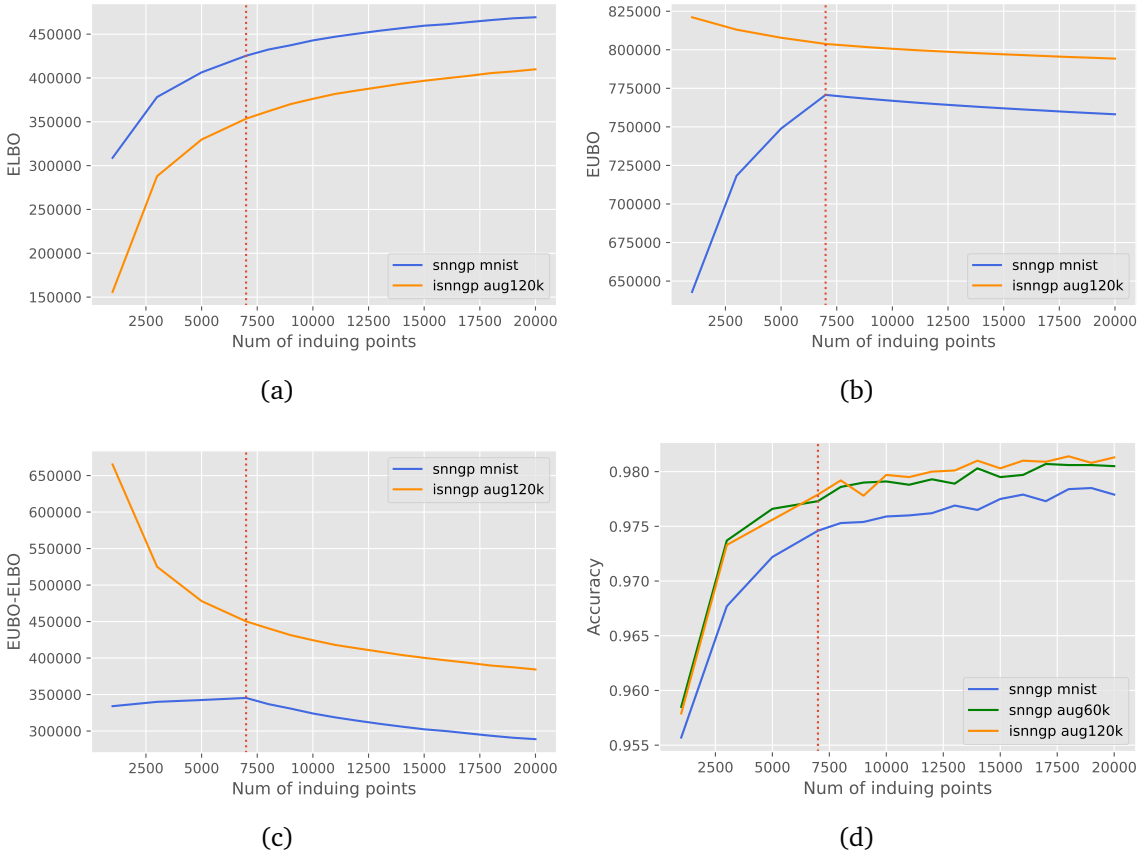


Figure 4.13: The performances of 3-layer infinitely wide neural networks with NNGP kernel and summation kernel on the MNIST. The test performance of standard data augmentation is also included in (d). Red vertical line indicates where the optimisation stops.

increasing inducing points on the iSNNGP contributes more to the convergence of the approximation rather than the improvement of data fitting. Also, the interval between boundaries of iSNNGP is much larger than that of the SNNGP, given the same size of kernel matrices.

The test accuracy of the iSNNGP verifies our expectation on the log marginal likelihood (Figure 4.13(d)). It significantly outperforms the SNNGP on the standard MNIST and achieves comparable accuracy to the SNNGP on the AUG60k. Surprisingly, the test performance of iSNNGP is slightly better than that of the SNNGP after 10000 inducing points.

4.6 Discussion

In the previous sections, we analyse the performance of sparse Gaussian processes with the NNGP kernel and the summation kernel on different sizes of datasets. We summarise our results from the following perspectives: 1) the quality of approxima-

tions as varying the number of inducing points, 2) the comparison between inducing points selections, and 3) the generalisation of models under different settings.

Given the curve fittings in Figure 4.5 and 4.3, the infinitely wide neural networks can fit the regions where the training data lies with high confidence. The finitely wide counterparts with much deeper layers show difficulty fitting the edge values. However, the infinitely wide neural networks predict extreme values at extrapolations and ultimately cause high test errors on average. Another benefit of using infinitely wide neural networks is that the overconfidence in unseen data can be avoided.

The sparse approximations of NNGPs on the synthetic and MNIST datasets demonstrate that the variational lower bound gradually converges to its target, the log marginal likelihood, while the target itself is improving as more inducing points. For two reasons, the interval between ELBO and EUBO does not reduce before the hyperparameters converge. First, the two boundaries change proportionally to the log marginal likelihood when M is small. Second, the EUBO takes a larger amount of inducing points to converge.

While the approximated distribution becomes close to the true posterior distribution within only 100 inducing points on the synthetic dataset, the greedy selection provides more informative inducing points that locate around the extreme points. Its marginal likelihood and boundaries converge much faster than those of random selection. Thus, the SNNGP with greedy selection requires a slightly smaller amount of inducing points to approximate than that with random selection.

In contrast to the results on the synthetic dataset, we can no longer use a small number of inducing points to approximate the true posterior unbiasedly. Every training sample carries information about the decision boundaries in the allowance of computational resources. The network always finds a better fitting as more inducing points are added. Moreover, the estimated likelihood variances of SNNGPs on the MNIST10k and AUG10k are 0.0001. Therefore, the models think that the noise in the MNIST dataset is very low. After we stop the optimisation of hyperparameters on the full MNIST dataset, the ELBO and EUBO slowly converge to the log marginal likelihood of the fixed model. But, there is still no evidence showing that the divergence between the approximated distribution and true posterior can be reasonably slight in at least 20000 inducing points.

Greedy selection does not significantly outperform the random selection on the MNIST dataset, especially when M is small, from both approximation and generalisation. We speculate that it is because the MNIST dataset is well-balanced and each sample carries a similar amount of information for its class. However, the greedy selection allows faster convergence of the approximated distribution to the exact posterior. As shown in Figure 4.7(e) and 4.11(d), faster convergence eventually leads to slightly robust test accuracy as increasing inducing points.

By adding deformed images, the tendencies of the boundaries of SNNGPs on the augmented dataset are precisely the same as those of the original ones, except the values are much higher due to larger kernel size. While the distance to the true

posterior is far because of larger values of approximate metrics, the test accuracy on the augmented dataset is higher than the accuracy on the standard dataset at each step. However, the major disadvantage of standard data augmentation is that the kernel size exponentially increases. So, it is difficult to optimise the approximate to get close to the true posterior on the augmented dataset with a moderate amount of inducing points.

On the other hand, the summation kernel avoids the above problem by treating the augmentations as the orbits of the original data. So, the kernel size remains the same. When the data is only doubled, the iSNNGP under-performs the SNNGPs on both standard MNIST and standard augmentations. The kernels of the iSNNGP on the doubled dataset are theoretically not different from the NNGP kernel between the inducing points and augmented data. But, its log marginal likelihood is smaller than that of the SNNGPs. Despite the kernel size being unchanged, an SNNGP still needs all training data (standard MNIST digits and augmentations) to approximate the exact posterior. So, each sample has a smaller amount of information on average as the total number of training data increases.

The ability of data fitting with the summation kernel seems to saturate much quicker than the models with the NNGP kernel. It causes the test performance of the iSNNGPs to be eventually lower than that of the SNNGPs (Figure 4.7(e)). This situation is evident in the smaller dataset. If we enlarge the size of orbits, the performance of the iSNNGP becomes better than the baseline when the number of inducing points is small. Nevertheless, rapid saturation still exists. From the empirical results, the iSNNGPs require a larger amount of augmentations than the SNNGP with standard augmentation to achieve similar test accuracy.

The kernel matrices are the most significant difference between the standard data augmentation and the learning invariances. Each sample has its own entry in a kernel matrix of the standard augmentation. The covariances between the augmentations and training data are explicit. In the summation kernel, the invariances are embedded in the entries of the original samples and become implicit. As more augmentations are produced, the invariant information in each entry is confusing. So, the performance of the iSNNGP on the larger augmentations only shows slight improvements.

Thus far, our results demonstrate that the test accuracy of the SNNGP on the MNIST10k dataset achieves 97.36%, which is much better compared with its finitely wide counterpart. However, the SNNGP on the MNIST dataset only performs 97.79% of test accuracy within 20000 inducing points and is around 0.3% lower than the finitely wide neural network. So, can we tune the hyperparameter to achieve similar test accuracy in 20000 inducing points regardless of the approximation bias? Following the results on the AUG10k, we can speculate the optimised σ_w, σ_b and likelihood variance if M exceeds 20000. By setting them to be 1.5, 0.1 and 0.0001, we obtain 98.46% of test accuracy with 20000 inducing points. But, the ELBO is -25794776.72, and the interval between boundaries is around 27503077.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this project, we implement the sparse Gaussian process regression for the infinitely wide neural networks in JAX. It overcomes the computational intractability of large matrix inverse and reduces the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$. We show that a close approximation to the true posterior can be achieved with fewer inducing points than the training data size on the highly noisy one-dimensional regression dataset. However, the sparse approximation on the MNIST dataset still requires a large number of inducing points and makes the computations difficult to accomplish on a single machine. Thus, sparse Gaussian process regression may not be an appropriate approximation method for classification. By comparing the random and greedy selection, the greedy selection provides more informative inducing points that lead to stable and faster convergence to the true posterior distribution for an approximated distribution.

Data augmentation is a simple method to improve the generalisation of networks. But, it is not efficient for Gaussian processes since the size of the kernel matrix increases exponentially. We implement the summation kernel by including the augmentations in each entry of the kernel and learning the invariances in a supervised manner. Empirical results show that the iSNNGPs need at least tripled training data to achieve similar test performance to the standard data augmentation with doubled training data. The summation kernel is surprisingly efficient at the low number of inducing points. However, the convergences of iSNNGPs saturate quickly. The test performance of iSNNGPs is exceeded by the SNNGPs with standard augmentation or even the original network, as increasing inducing points.

5.2 Future Work

Since the sparse Gaussian process regression assumes the Gaussian likelihood and we have shown that this method is unsuitable for the MNIST dataset, our implementations can be further investigated in two ways. First, we can optimise the objective

function using stochastic optimisations to minimise the KL divergence between the approximate and true posterior (37). The new objective function (ELBO) is given by

$$\log p(\boldsymbol{y}) \geq \mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{f})}[\log p(\boldsymbol{y}|f(\boldsymbol{x_n}))] - \mathbf{KL}[q(\boldsymbol{u})||p(\boldsymbol{u})] \tag{5.1}$$

where $\boldsymbol{u}$ is the function values of the inducing points. The computational cost, in this case, reduces to $\mathcal{O}(BM^2+M^3)$ where B is the batch size. Second, we can remove the constraint on the Gaussian likelihood and use non-conjugate likelihoods (59), such as Bernoulli likelihood for binary classification and multinomial likelihood for multi-class classification. With the summation kernel, the above ELBO can be evaluated either analytically or by Monte Carlo. So, we obtain a technique that allows Gaussian processes on large datasets with general likelihoods.

We can further improve the memory cost of large matrix computations from an engineering perspective. The matrix inverse only needs $O(N^2)$ memory space if the calculation is in-place. Theoretically, the inverse of the full kernel matrix on the MNIST dataset requires around 28 gigabytes using 64-bit floating points. In practice, inversing a 60000×60000 matrix is not possible in either NumPy or JAX on our experimental platform. Since a kernel matrix is symmetric and positive semidefinite, we attempt to inverse the matrix in blocks by Schur complement. However, the memory usage is still too large to finish the computation. The other work (25) uses the SciPy and overwrites the original matrix. But, JAX does not support overwriting due to immutability and using SciPy requires moving the matrix between packages and devices. So, a memory-efficient matrix inverse needs to be further investigated and developed.

Bibliography

[1] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL https://proceedings.mlr.press/v5/titsias09a.html. pages i, 11, 12, 13, 15

[2] Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/d465f14a648b3d0a1faa6f447e526c60-Paper.pdf. pages i, 17, 18

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385. pages 1

[4] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016. URL https://arxiv.org/abs/1608.06993. pages 1

[5] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, page 473–479, Cambridge, MA, USA, 1996. MIT Press. pages 1

[6] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014. URL https://arxiv.org/abs/1409.1259. pages 1

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish,

Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165. pages 1

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. URL https://arxiv.org/abs/2010.11929. pages 1

[9] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. URL https://arxiv.org/abs/2205.11487. pages 1

[10] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208. pages 1

[11] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274. pages 1

[12] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T. URL https://www.sciencedirect.com/science/article/pii/089360809190009T. pages 1

[13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/guo17a.html. pages 1

[14] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations, 2019. URL https://arxiv.org/abs/1903.12261. pages 1

[15] Radford M. Neal. Bayesian learning for neural networks. 1995. pages 2, 8, 10

[16] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes, 2017. URL https://arxiv.org/abs/1711.00165. pages 2, 8, 9, 10

[17] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes, 2018. URL https://arxiv.org/abs/1810.05148. pages 2, 10

[18] Greg Yang. *Tensor Programs I: Wide Feedforward or Recurrent Neural Networks of Any Architecture Are Gaussian Processes*. Curran Associates Inc., Red Hook, NY, USA, 2019. pages 2, 10

[19] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005. ISBN 9780262256834. doi: 10.7551/mitpress/3206.001.0001. URL https://doi.org/10.7551/mitpress/3206.001.0001. pages 5, 12, 13

[20] David John Cameron Mackay. *Bayesian Methods for Adaptive Models*. PhD thesis, USA, 1992. UMI Order No. GAX92-32200. pages 8

[21] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, may 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448. URL https://doi.org/10.1162/neco.1992.4.3.448. pages 8

[22] Wray L. Buntine and Andreas S. Weigend. Bayesian back-propagation. *Complex Syst.*, 5(6), 1991. URL http://www.complex-systems.com/abstracts/v05_i06_a04.html. pages 8

[23] Alexander G. de G. Matthews, Mark Rowland, Jiri Hron, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks, 2018. URL https://arxiv.org/abs/1804.11271. pages 9, 10

[24] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/file/5751ec3e9a4feab575962e78e006250d-Paper.pdf. pages 9

[25] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes, 2018. URL https://arxiv.org/abs/1808.05587. pages 10, 22, 39

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. pages 10

[27] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017. pages 10

[28] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks, 2019. URL https://arxiv.org/abs/1901.11196. pages 10

[29] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data, 2015. URL https://arxiv.org/abs/1511.06709. pages 10

[30] Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-1083. pages 10

[31] Matthias W. Seeger, Christopher K. I. Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In Christopher M. Bishop and Brendan J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, volume R4 of *Proceedings of Machine Learning Research*, pages 254–261. PMLR, 03–06 Jan 2003. URL https://proceedings.mlr.press/r4/seeger03a.html. Reissued by PMLR on 01 April 2021. pages 11

[32] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005. URL http://jmlr.org/papers/v6/quinonero-candela05a.html. pages 11

[33] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2005/file/4491777b1aa8b5b32c2e8666dbe1a495-Paper.pdf. pages 11

[34] Felix Leibfried, Vincent Dutordoir, ST John, and Nicolas Durrande. A tutorial on sparse gaussian processes and variational inference, 2020. URL https://arxiv.org/abs/2012.13962. pages 12

[35] Alexander G. de G. Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the kullback-leibler divergence between stochastic processes. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 231–239, Cadiz, Spain, 09–11 May 2016. PMLR. URL https://proceedings.mlr.press/v51/matthews16.html. pages 13

[36] Michalis K Titsias. Variational inference for gaussian and determinantal point processes. In *Workshop on Advances in Variational Inference (NIPS)*, 2014. pages 14, 23

[37] James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 282–290, Arlington, Virginia, USA, 2013. AUAI Press. pages 14, 17, 39

[38] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017. URL `http://jmlr.org/papers/v18/16-537.html`. pages 14

[39] Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv:2003.01115*, 2020. URL `https://arxiv.org/abs/2003.01115`. pages 14

[40] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL `https://github.com/google/neural-tangents`. pages 14, 20

[41] Thomas Pinder and Daniel Dodd. Gpjax: A gaussian process framework in jax. *Journal of Open Source Software*, 7(75):4455, 2022. doi: 10.21105/joss.04455. URL `https://doi.org/10.21105/joss.04455`. pages 15

[42] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. pages 15

[43] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989. pages 15

[44] Laming Chen, Guoxin Zhang, and Eric Zhou. Fast greedy map inference for determinantal point process to improve recommendation diversity. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/dbbf603ff0e99629dda5d75b6f75f966-Paper.pdf`. pages 15

[45] Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975. ISSN 00018678. URL `http://www.jstor.org/stable/1425855`. pages 15

[46] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors,

Advances in Neural Information Processing Systems, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/0eec27c419d0fe24e53c90338cdc8bc6-Paper.pdf. pages 15

[47] Alex Kulesza and Ben Taskar. Learning determinantal point processes. 2011. pages 15

[48] Alex Kulesza and Ben Taskar. K-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 1193–1200, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195. pages 15

[49] Alex Kulesza. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286, 2012. doi: 10.1561/2200000044. URL https://doi.org/10.1561%2F2200000044. pages 15

[50] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978. pages 15

[51] David R. Burt, Carl Edward Rasmussen, and Mark van der Wilk. Convergence of sparse variational inference in gaussian processes regression. *Journal of Machine Learning Research*, 21(131):1–63, 2020. URL http://jmlr.org/papers/v21/19-1015.html. pages 15

[52] Miguel Lázaro-Gredilla and Aníbal Figueiras-Vidal. Inter-domain gaussian processes for sparse inference using inducing features. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/file/5ea1649a31336092c05438df996a3e59-Paper.pdf. pages 18

[53] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. pages 19

[54] Yann Lecun, Leon Bottou, Genevieve Orr, and Klaus-Robert Müller. Efficient backprop. 08 2000. pages 20

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. pages 20

[56] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html. pages 21

[57] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018. pages 21

[58] Jascha Sohl-Dickstein, Roman Novak, Samuel S. Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *CoRR*, abs/2001.07301, 2020. URL `https://arxiv.org/abs/2001.07301`. pages 21

[59] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015. pages 39