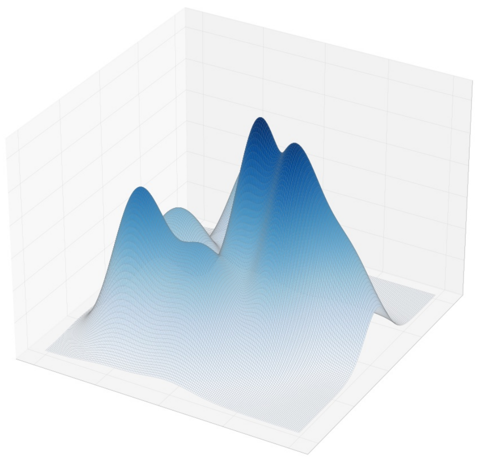
Function Space Transfer of Probability Distributions



Georges Pantalos

July 20, 2021

Master Thesis in the Department of Mechanical Engineering at the Learning & Adaptive Systems Group, Machine Learning Institute, ETH Zürich.

To my grandfather — Georgios Pantalos

Acknowledgments

I would like to thank my supervisor Jonas Rothfuss, for introducing me to Bayesian deep learning and providing invaluable feedback to my work both in terms of guidance and fundamental intuition. I would also like to thank the Computer Science department at ETH Zurich, for providing computational resources that made this thesis possible. Finally, I would like to thank professor Andreas Krause, for practical advice on general aspects of scientific research.

Abstract

Building accurate models of dynamical systems is essential for control applications. System identification approaches hand-craft prior knowledge – derived from physical principles in the form of differential equations – into restrictive parametric models that can be arbitrarily inaccurate because of confining assumptions. Deep learning approaches on the other hand are very expressive but lack the ability to represent prior knowledge efficiently due to their complex and high-dimensional parameter space. Therefore, most prior knowledge is discarded.

We propose a method combining the best of both worlds. Namely, we use the data-generating process as an implicit prior in the function space to train Bayesian neural networks. We demonstrate the performance of these models on a range of regression tasks and compare and contrast with related models in the literature.

Contents

List of Figures

Nomenclature

General

$\mathbf{X}$	matrix
$\mathbf{x}$	vector
x	scalar

Calligraphic

$\mathcal{A}$	Stein operator
$\mathcal{D}$	dataset
$\mathcal{F}$	function space
$\mathcal{H}$	reproducing kernel Hilbert space
$\mathcal{L}$	log-normal distribution
$\mathcal{N}$	normal distribution
$\mathcal{Q}$	variational family of distributions
$\mathcal{U}$	uniform distribution
$\mathcal{X}$	input space
$\mathcal{Y}$	output space

Blackboard

$\mathbb{N}$	natural numbers
$\mathbb{R}$	real numbers
$\mathbb{S}$	Stein discrepancy

Distributions

p	prior
q	posterior
$q_{\boldsymbol{\theta}}$	variational posterior parameterized by $\boldsymbol{\theta}$
$\nu_{\boldsymbol{\omega}}$	data-generating process parameterised by $\boldsymbol{\omega}$

Other

Pr	probability density
E_p	expectation with respect to distribution p
$\mathbf{K}$	gram matrix
$\mathbf{W}$	weight matrix
$\mathbf{w}$	vectorized neural network parameters

Notation

Throughout this thesis, we refer to models that benefit from a function space treatment of the predictive posterior distribution during training as *function-space models* or *functional models* in contrast to models that only explicitly deal with parameters in the weight space.

Chapter 1

Introduction

Consider the problem of building accurate models of mechanical robots. That is, building a function which describes how the robot will evolve over time as a stochastic function of the current state and actions. While most systems are well understood and can be described by equations of motion deriving from physical considerations, most of the resulting models are too simplistic and rely on confining assumptions. Furthermore, some parameters (e.g. friction coefficients and moments of inertia) are usually unknown and non-trivial to infer.

A field that has attempted to solve the latter is *system identification*. It has its origins in the 19th century with the works of Gauss (1809) and Fisher and saw an increase in popularity in the mid 1900s when it was applied to engineering problems (Ho and Kalman, 1965). However, this approach is often too restrictive to capture the full complexity of mechanical interactions as it does not address the issue of having flawed assumptions about the model, also known as *epistemic uncertainty*. In many cases, further precision is required.

Traditional deep learning algorithms try to estimate the parameters of a highly complex and nonlinear model that best explain a set of observations under a certain metric. This time, the issue relies on the high variance of this approach, as deep models are often too expressive and tend to *overfit* to the training data. An important drawback of traditional (in the sense that they are non-Bayesian) neural networks is that they only provide *point estimates* of the predictions. They can account for *aleatoric uncertainty* which arises from noisy data due to measurement imprecision or other physical factors. However, they fail to capture epistemic uncertainty (or model uncertainty) consisting of uncertainty in model parameters (arising from the different combinations of parameters explaining the same data) and model structure.

Bayesian machine learning models address this issue by providing a distribution over the possible values of the parameters and can therefore better capture uncertainty in model parameters, a problem known as *model selection*. A central task in these methods is the evaluation of the *posterior distribution* density $\Pr(\theta \mid \mathcal{D})$ of latent variables θ given observations $\mathcal{D}$. However, because of the intractability the model evidence in the marginalization of model parameters (explained analytically later in this introduction), Bayesian methods applied to neural networks are often reduced to representing the posterior using a diagonal covariance Gaussian which is far too restrictive (MacKay, 1992; Neal, 1992). Nevertheless, the Bayesian framework allows for the introduction of prior information, that is, any information believed to be relevant to the task at hand in the form of a *prior distribution* over model parameters.

The idea behind Bayesian modeling is the following. The prior covers a large set of functions, but only a few of these functions are compatible with the data, allowing the posterior to concentrate onto a smaller set of functions, all of which are likely to be good. Ideally, the prior should be chosen in a way such that it accurately reflects our beliefs about the parameters before seeing any data (Gelman, 1996).

1.1 Problem Statement

In Bayesian modeling, usually a domain expert hand-crafts custom prior distributions over the model parameters that reflect his or her prior beliefs. However, this is hard to do for neural networks due to their complex and high-dimensional parameter space. Therefore in Bayesian deep learning, it is common practice to choose a prior distribution over parameters that is *uninformative* (or weakly informative), such as an isotropic standard Gaussian, a concept inspired by theoretical results in Bayesian statistics (Vaart, 1998). By uninformative, we mean that it does not contain information that is more relevant to the task at hand than any other task, i.e. it does not reflect our beliefs about the data. The tendency of choosing such a prior is caused by the lack of alternative solutions for efficient evaluation of the prior density. This raises the central question of my thesis.

Can we represent this prior information as a probability distribution?

More precisely, can we cast information that is available in the function space as a distribution over parameters in the weight space? For example in the case of modeling the dynamics of a mechanical robot, can we represent physical properties of the robot and what we know about its interaction with the environment (e.g. equations of motion) as a distribution over the parameters of our model?

As mentioned earlier, deep Bayesian models tend to use non-informative priors by lack of alternative efficient solutions. However, the posterior can be arbitrarily bad if the data-generating function is not in the set of functions covered by the prior. It therefore makes sense to think of the prior as whatever information we have about the task before gathering data. This distinction is at the heart of our reasoning for finding better priors for Bayesian models.

A practical approach is *meta-learning* the prior. By meta-learning, we mean learning from data issued from different tasks that share common characteristics (e.g. solutions of the ODEs describing the dynamics of a robot for different parameter configurations as explained in the introduction of this thesis) and meta-learning the best hyper-parameters of the explicit prior distribution. While this approach conveys information about the task to the prior, it is still too restrictive to model a wide array of inductive biases since the prior is again represented by factorized Gaussian distributions for efficiency purposes.

Our work follows from this approach of explicitly meta learning the prior. In this setting, we use the equivalent of samples from meta datasets during training time in order to use the approximate data-generating process as an implicit prior in function space and make the posterior tractable. More specifically, we use the main result from Sun et al. (2019) that allows the computation of a KL divergence between stochastic processes and extend it to learn an implicit prior process. We experiment with different randomness settings in the generation of the meta data and compare our work to Rothfuss et al. (2021); Liu and Wang (2019); Sun et al. (2019); Wang et al. (2019). Formally, we compare two functional models, namely FVI (functional variational inference) and FSVGD (functional Stein variational gradient descent), to two their weight-space counterparts, respectively VI and SVGD. We also use a Bayesian neural network with a meta-learned prior denoted PACOH. For the exact description of these models we direct the reader to chapters 3 and 4 and for the PACOH model to Rothfuss et al. (2021). Bear in mind that functional models have *oracle access* to the data-generating distribution. They can draw samples from it infinitely during training in contrast to PACOH which is designed to have access to a limited number of tasks during meta-training.

1.2 Main Takeaways

The conducted experiments at different scale and complexity levels indicate that functional models outperform non-functional ones according to both negative log likelihood (NLL) and root mean squared error (RMSE). These include data generating processes derived from sinusoidal functions, mixtures of normal densities and environments from the physics simulator MuJoCo which provide a diverse range of dynamics and improves the generalizability of the results.

The other more general takeaway is that functional models (i.e. Bayesian neural networks that use the data-generating process as a prior in function space) bridge the gap between approaches that use expert domain knowledge to construct a prior but are too restrictive and very expressive approaches which cannot make efficient use of this prior knowledge because of their high-dimensional and complex parameter space.

1.3 Thesis Outline

The thesis is structured as follows. Chapter 2 is concerned with formalizing the problem and presenting the historical context in detail. In chapters 3 and 4, we explain the methods involving weight space and, respectively, function space method of handling of model uncertainty and the intuition behind them. Later, in chapter 5, we explain in detail how the experiments were conducted and display the results of our study. Ultimately, in chapter 6 we discuss application domains and possible improvements of our method.

Chapter 2

Background: Bayesian Neural Networks

In this chapter, we present the general framework of Bayesian inference and how it applies to neural networks. Then we present the problem of intractability in exact Bayesian inference and cite alternative approximation schemes. We then give a short historical overview of Bayesian neural networks for the interested reader.

2.1 The Intractability of Bayesian Inference

Consider a model parameterized by $\boldsymbol{\theta}$ and a dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{i=N}$ containing N tuples of labeled observations. We assume for the sake of simplicity that observations and labels are scalar real numbers, that is: $x_i, y_i \in \mathbb{R},\ \forall i$. The exact computation of the posterior as a closed form expression is non-trivial. Following Bayes' rule, it involves the computation of the prior $\Pr(\boldsymbol{\theta})$, the likelihood $\Pr(\mathcal{D} \mid \boldsymbol{\theta})$ and another term known as model evidence or partition function $\Pr(\mathcal{D})$. Indeed, the logarithmic version of Bayes theorem is

$$\log \Pr(\theta \mid \mathcal{D}) = \log \Pr(\mathcal{D} \mid \theta) + \log \Pr(\theta) - \log \Pr(\mathcal{D}) \tag{2.1}$$

The first two terms are in general straightforward to compute. Assuming that data is identically and independently distributed (i.i.d.), the data log-likelihood can be written as a sum of log-likelihoods of individual target values y_i in the dataset under the predictive distribution, e.g. a forward pass in a neural network, as such

$$\log \Pr(\mathcal{D} \mid \theta) = \log \prod_{i=1}^{N} \Pr(y_i \mid x_i, \theta) = \sum_{i=1}^{N} \log \Pr(y_i \mid x_i, \theta) \tag{2.2}$$

Nonetheless, the efficiency of the prior term computation strongly depends on the choice of distribution selected. It is usually not a computational bottleneck as we only consider the problem of evaluating the posterior for one configuration of the parameters θ. The remaining term is usually causing the intractability in the computation of the posterior. Indeed, it can only result from marginalisation of all the model parameters,

$$\Pr(\mathcal{D}) = \int \Pr(\mathcal{D} \mid \theta) \Pr(\theta) \mathrm{d}\theta \tag{2.3}$$

For completeness, let us elaborate on why exactly it is that this integral is intractable and therefore, why it is useful to approximate it. Note that the parameter vector $\boldsymbol{\theta} := [\theta_1, \ldots, \theta_n]$ where $\theta_i \in \mathbb{R}$ is often high-dimensional, leading to the computation of a tedious multiple integral of the form

$$\int_{\theta_1} \cdots \int_{\theta_n} \Pr(\mathcal{D} \mid \boldsymbol{\theta}) \Pr(\boldsymbol{\theta}) \; \mathrm{d}\theta_1 \ldots \mathrm{d}\theta_n \tag{2.4}$$

with n being in the range of several millions for large neural networks. Because of this intractable integral, the exact posterior cannot be evaluated analytically. One could think of approximating the integral using Monte Carlo methods. However these too often fail to scale to higher dimensions. Since exact Bayesian inference is not feasible in models involving a large number of parameters, we resort to approximation schemes.

Prediction In Bayesian machine learning, the framework is slightly different for training and prediction. When making predictions, we ought to marginalise out the parameters. The predictive distribution for a test input x^* is

$$\Pr(x^* \mid \mathcal{D}) = \int \Pr(x^* \mid \mathcal{D}, \boldsymbol{\theta}) \;\; \Pr(\boldsymbol{\theta} \mid \mathcal{D}) \; \mathrm{d}\boldsymbol{\theta} \tag{2.5}$$

This process is known as *inference* in Bayesian modelling. This is in contrast to standard maximum-likelihood approaches used in frequentist models where only one parameter vector $\boldsymbol{\theta}^* := \arg\max_{\boldsymbol{\theta}} \Pr(\mathcal{D} \mid \boldsymbol{\theta})$ is used:

$$\Pr(x^* \mid \mathcal{D}) = \Pr(x^* \mid \mathcal{D}, \boldsymbol{\theta}^*) \tag{2.6}$$

2.2 Approximate Inference

Looking to resolve this issue, the field of approximate inference has earned a lot of attention in recent years. There are two subcategories of approximation schemes: *stochastic* and *deterministic*. The former has the property that using an infinite amount of samples it can generate exact results – examples include the popular Markov Chain Monte Carlo (MCMC) methods as in Neal (1995); Chen et al. (2000). Deterministic approximation schemes are concerned with approaching the posterior analytically which trades off precision for computational efficiency. We will concentrate on the latter in this thesis.

2.3 The Bayesian Neural Network

Consider a linear neural network layer with weight matrix $\mathbf{W} \in \mathbb{R}^{n \times m}$, bias vector $\mathbf{b} \in \mathbb{R}^m$ and a single input $\mathbf{x} \in \mathbb{R}^n$. The output of the layer is computed as follows $\mathbf{y} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$. In the Bayesian case, we introduce uncertainty in model parameters. We sample the weights and biases from a posterior distribution and update the parameters of that distribution. Suppose that we gather all the parameters of the neural network in a vector $\mathbf{w}$ and define a distribution $q_{\boldsymbol{\theta}}(\mathbf{w})$ over it parameterised by a vector $\boldsymbol{\theta}$. The parameters that are optimized are now the arguments of the posterior instead of the weights and biases of the neural network.

Remark. Note that here and for the rest of this thesis, $\boldsymbol{\theta}$ represents the parameters of the (variational) posterior. This is in contrast to §2.1 where it represents the model parameters (e.g. the weights of a neural network).

Sampling methods for updating $\boldsymbol{\theta}$ are reliable but computationally expensive. These involve obtaining gradient estimates through Monte Carlo sampling (Mohamed et al., 2020) and using them to backpropagate the loss through the parameters of the posterior. A more efficient approach consists of choosing a variational family of distributions and finding the element that maximises a specific bound called the ELBO (more on this in section 3.1). The predictions are done according to (2.5).

2.4 A Short History of BNNs

We can trace the origin of Bayesian neural networks to when Denker et al. (1987) proposed placing a prior over weights and mapping each weight configuration to a set of network outputs. This permitted marginalisation of the weights to obtain a predictive distribution. Following that work, Denker and LeCun (1990) suggested using Laplace's method to approximate the posterior over neural network parameters. A few years later, MacKay (1992) introduced the concept of using model evidence for model selection as it correlates with model generalisation. Then, variational inference came in the picture with the work of Hinton and van Camp (1993) in using minimum description length principles to penalise excess information in the model parameters. In his PhD thesis, Neal (1995) worked on inference approximations schemes based on Monte Carlo sampling and developed the *Hamiltonian Monte Carlo* (HMC) algorithm, which is still considered the gold standard in terms of accuracy in Bayesian neural networks to this day. He also showed that Bayesian neural networks with one hidden layer of infinite width converge to a Gaussian process when a Gaussian prior is used. Ultimately, Barber and Bishop (1998) built upon Hinton's work and replaced the diagonal covariance matrices with full covariance matrices, making the model more expressive. Additionally they called attention to the fact that the training objective function formed a lower bound to the model evidence, a quantity known nowadays as *evidence lower bound*, or ELBO.

More recently, Graves et al. (2011) proposed a practical method for variational inference with fully factorised Gaussian posteriors using data sub-sampling (i.e. minibatch optimization) to account for the computational efficiency loss induced by the non-sparsity of the posterior. Improving on that work, Blundell et al. (2015) used the *reparameterization trick* proposed earlier by Kingma and Welling (2014) to sample expected log-likelihood estimates and suggested using a mixture of Gaussians prior over each weight to improve model performance. In his PhD thesis, Gal (2016) showed that classical neural networks with a *Dropout* layer can also be interpreted as Bayesian neural networks. Louizos and Welling (2017) use normalising flows to construct a non-factorised, non-Gaussian approximate posterior in Bayesian neural networks. Wen et al. (2018) introduced the *Flipout* layer that casts the activations as stochastic variables (instead of the weights) to reduce variance during training. However, none of these methods make use of uncertainties in function space directly.

Chapter 3

Weight-Space View

The most accurate techniques for Bayesian inference in neural networks usually involve sampling methods. However, these approaches often become intractable in higher dimensions. A more efficient method involving restricting the posterior to a variational family of distributions and minimizing a discrepancy to keep the posterior close to the prior is known as *variational inference*. In this section we review the weight space treatments of approximate inference as well as the models that will be used as baselines for our experiments.

3.1 Variational Inference

Variational methods were introduced in the late 1700s by German and French mathematicians Leonhard Euler and Joseph-Louis Lagrange with their work on the *calculus of variations*. Attempting to model the posterior in parameter-space in BNNs using variational inference was initially presented in Graves et al. (2011). It is approximated by minimizing a discrepancy between the true posterior and a variational family of distributions (e.g. isotropic Gaussians).

3.1.1 Evidence Lower Bound

Formally, let us denote by $p(\mathbf{w})$ a prior distribution over vectorised neural network parameters $\mathbf{w}$, by $q(\mathbf{w})$ a variational posterior and by $\mathcal{Q}$ a family of variational posteriors over $\mathbf{w}$. For example, $\mathcal{Q}$ could be the set of multivariate normal distributions with diagonal covariance matrices. The resulting training objective can be formulated as a combination of maximizing the expected likelihood and minimizing the distance between prior and variational posterior distributions,

$$\min_{q_{\boldsymbol{\theta}} \in \mathcal{Q}} \mathop{\mathrm{E}}_{\mathbf{w} \sim q_{\boldsymbol{\theta}}} \left[-\log \Pr(\mathcal{D} \mid \mathbf{w}) - \log p(\mathbf{w}) + \log q(\mathbf{w}) \right] \tag{3.1}$$

This optimization objective is known as the *evidence lower bound* (ELBO). The first term denotes the data likelihood and makes sure the parameters are such that the model fits the training data. The second ensures that the sampled weights are close to the prior and the third introduces entropy in the parameters, i.e. ensures that different parameters are sampled each time. Note that the two rightmost terms form the *Kullback–Leibler divergence* between distributions q and p,

$$\mathrm{KL}(q, p) = \mathop{\mathrm{E}}_{\mathbf{w} \sim q} \log \frac{q(\mathbf{w})}{p(\mathbf{w})} \tag{3.2}$$

This procedure of framing a Bayesian inference problem as an optimization problem is known as *variational inference*.

3.1.2 Mean Field Approximation

A computationally convenient approximation is to assume that the joint distribution over the weights can be factorized. It is particularly favorable since the log density of the posterior can be computed as a simple sum of logarithms

$$q(\mathbf{w}) = \prod_{i=1}^{n} q(w_i) \implies \log q(\mathbf{w}) = \sum_{i=1}^{n} \log q(w_i) \tag{3.3}$$

This is known as the *mean field approximation*.

> **Remark.** The variational posterior is, in general, parametric in which case we denote it by $q_{\boldsymbol{\theta}}$ with parameter vector $\boldsymbol{\theta}$.

3.2 Stein Variational Gradient Descent

Another approach for approximating the posterior is a member of the so-called particle-optimization-based variational inference (POVI) methods as explained in Liu and Wang (2019). This method can be seen as gradient descent on a set of particles and is known as *Stein Variational Gradient Descent* (SVGD). The main idea is to compute the kernel matrix between the particles and use it to keep the particles reasonably far from each other while iteratively moving them closer to the mode. In this section, we will roughly outline the main ideas behind the SVGD algorithm including the Stein identity and kernelized Stein discrepancy.

A useful preliminary notion for this chapter is the *score function*.

Definition 3.1 (Score function)**.** Consider a density p. The score function $g(x)$ of $p(x)$ is defined as the gradient of its log-density with respect to the input dimension. That is,

$$g(x) = \nabla_x \log p(x) \tag{3.4}$$

3.2.1 Stein's Identity

Recent developments in Stein discrepancy and its kernelized extensions (Gorham and Mackey, 2015) have renewed the interests in Stein's method, which is a classic tool in statistics. Central to these works is an equation that generalizes the original Stein's identity (Stein, 1981) which is the foundation of this method and displayed hereafter. First, a preliminary definition.

Definition 3.2 (Stein class)**.** A function f is said to be in the *Stein class* of a probability measure p if

$$\int_{\mathbf{x}\in\mathcal{X}} \nabla_{\mathbf{x}} \left[f(\mathbf{x})p(\mathbf{x})\right] \mathrm{d}\mathbf{x} = 0 \tag{3.5}$$

Theorem 3.1 (Stein's Identity)**.** *Let p be a smooth density and $\boldsymbol{\phi}(x) := [\phi_1(x), \ldots, \phi_d(x)]^\top$ a smooth vector function with all its elements ϕ_i in the Stein class of p. Then,*

$$\mathop{\mathrm{E}}_{x\sim p}\left[\mathcal{A}_p \boldsymbol{\phi}(x)\right] = 0 \tag{3.6}$$

where $\mathcal{A}_p$ *is the* Stein operator *defined by* $\mathcal{A}_p\boldsymbol{\phi}(x) := \boldsymbol{\phi}(x)\nabla_x \log p(x)^\top + \nabla_x \boldsymbol{\phi}(x)$.

Let's use a simple example to illustrate this result.

Example 3.1 (Stein's Identity)**.** Consider a density $p = \mathcal{N}(0,1)$ and function $\boldsymbol{\phi}(x) = [x, x^2]^\top$. Then, the gradient of the vector function is $\nabla_x \boldsymbol{\phi}(x) = [1, 2x]^\top$ and the score function $\nabla_x \log p(x) = -x$. Accordingly, $\mathcal{A}_p\boldsymbol{\phi}(x) = -x[x, x^2]^\top + [1, 2x]^\top = [-x^2+1, -x^3+2x]^\top$ which gives the following expectation

$$\mathop{\mathrm{E}}_{x\sim p}[\mathcal{A}_p\boldsymbol{\phi}(x)] = \int_{\mathbb{R}} \begin{bmatrix} -x^2+1 \\ -x^3+2x \end{bmatrix} p(x)\mathrm{d}x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

because $(-x^3+2x)p(x)$ is an odd function — the rest can be solved by parts as well as making use of the Gaussian integral, $\int_{\mathbb{R}} \exp(-x^2) = \sqrt{\pi}$. △

3.2.2 Stein Discrepancy

If we take samples from a different smooth density q, Stein's identity does not hold anymore, meaning that $\mathrm{E}_{x\sim q}[\mathcal{A}_p\boldsymbol{\phi}(x)]$ is not necessarily equal to zero. This means that it can be used to describe a discrepancy measure between distributions p and q, known as the *Stein discrepancy*

$$\mathbb{S}(q,p) := \max_{\boldsymbol{\phi}\in\mathcal{F}} \left\{ \left[\mathop{\mathrm{E}}_{x\sim q} \operatorname{tr} \mathcal{A}_p\boldsymbol{\phi}(x) \right]^2 \right\} \tag{3.7}$$

However, it is non-trivial to choose a convenient function space $\mathcal{F}$ leading to a tractable optimization problem.

3.2.3 Kernelized Stein Discrepancy

A handy way to do this arises from reproducing kernel Hilbert spaces (RKHS). If we choose $\mathcal{F}$ to be the unit ball of a d-dimensional RKHS which we denote $\mathcal{H}^d$ induced by a kernel $k(\cdot,\cdot)$, the optimization problem in (3.7) can be expressed as

$$\mathbb{S}(q,p) = \max_{\boldsymbol{\phi}\in\mathcal{H}^d} \left\{ \left[\mathop{\mathrm{E}}_{x\sim q} \operatorname{tr} \mathcal{A}_p\boldsymbol{\phi}(x) \right]^2, \text{ such that } \|\boldsymbol{\phi}\|_{\mathcal{H}^d} \leq 1 \right\} \tag{3.8}$$

and is called the *kernelized Stein discrepancy*. The reason this formulation is so convenient is that the optimization problem in (3.8) has a closed form solution

$$\boldsymbol{\phi}(x) = \frac{\boldsymbol{\phi}^{\star}_{q,p}(x)}{\left\|\boldsymbol{\phi}^{\star}_{q,p}\right\|_{\mathcal{H}^d}} \tag{3.9}$$

where $\boldsymbol{\phi}^{\star}_{q,p}(\cdot) = \mathrm{E}_{x\sim q}\left[\mathcal{A}_p k(x,\cdot)\right]$ and produces a value for the KSD of $\mathbb{S}(q,p) = \left\|\boldsymbol{\phi}^{\star}_{q,p}\right\|^2_{\mathcal{H}^d}$. Additionally, note that $\mathbb{S}(q,p) = 0 \iff p = q$. The main difference between KSD and other metrics between distributions is that it depends on p only through the score function which does not depend on the normalization constant. The concepts of score function and unnormalized densities will be further investigated in §4.2.

3.2.4 Smooth Transforms

The next step is to choose a variational family $\mathcal{Q}$ to perform variational inference in our model as explained earlier (§3.1). In order to balance *accuracy*, *tractability* and *solvability* of the variational posterior, we can restrict it to a set consisting of distributions obtained by smooth transforms from a tractable reference distribution. That is,

$$\mathcal{Q} = \{q(z) : z = \boldsymbol{T}(x)\} \tag{3.10}$$

where $\boldsymbol{T} : \mathcal{X} \to \mathcal{X}$ is a smooth bijection and x is drawn from a tractable reference distribution $q_0(x)$. The density of the transformed random variable z is can be computed using the change of variable formula

$$q_{[\boldsymbol{T}]}(z) = q\left(\boldsymbol{T}^{-1}(z)\right) \cdot \left|\det\left(\nabla_z \boldsymbol{T}^{-1}(z)\right)\right| \tag{3.11}$$

where $\boldsymbol{T}^{-1}$ denotes the inverse map of $\boldsymbol{T}$ and $\nabla_z \boldsymbol{T}^{-1}$ the Jacobian matrix of $\boldsymbol{T}^{-1}$. Such distributions are computationally tractable, in the sense that the expectation under $q_{[\boldsymbol{T}]}$ can be easily evaluated by averaging samples $\{z_i\}$ when $z_i = \boldsymbol{T}(x_i)$ and $x_i \sim q_0$. The only problem that remains is to restrict the set of transforms $\boldsymbol{T}$ to make the optimization in (3.1) solvable. It turns out we can iteratively construct incremental transforms that effectively perform steepest descent on $\boldsymbol{T}$ in RKHS. The main result we will be using is the following.

Theorem 3.2 (Stein Operator as the Derivative of KL Divergence)**.** *Consider a transform* $\boldsymbol{T}(x) = x + \epsilon\boldsymbol{\phi}(x)$ *and* $q_{[\boldsymbol{T}]}(z)$ *the density of* $z = \boldsymbol{T}(x)$ *when* $x \sim q(x)$. *Then*

$$\nabla_\epsilon \operatorname{KL}(q_{[\boldsymbol{T}]} \| p)\big|_{\epsilon=0} = - \operatorname*{E}_{x\sim q} \operatorname{tr} \mathcal{A}_p \boldsymbol{\phi}(x) \tag{3.12}$$

We now have a direct connection between the Stein operator and the derivative of KL divergence with respect to the perturbation magnitude ϵ. Following from Theorem 3.2, we can infer that the direction of steepest descent $\boldsymbol{\phi}^*$ that minimizes the KL gradient in (3.12) is

$$\boldsymbol{\phi}^*_{q,p}(\cdot) = \operatorname*{E}_{x\sim q}\left[k(x,\cdot)\nabla_x \log p(x) + \nabla_x k(x,\cdot)\right] \tag{3.13}$$

This suggests an iterative procedure that transforms an initial reference distribution q_0 to the target distribution p. We construct a path of distributions $\{q_\ell\}_{\ell=1}^n$ such that

$$q_{\ell+1} = q_{\ell[\boldsymbol{T}^*_\ell]}, \qquad \text{where} \qquad \boldsymbol{T}^*_\ell(x) = x + \epsilon_\ell \cdot \boldsymbol{\phi}^*_{q_\ell,p}(x) \tag{3.14}$$

This procedure converges in the limit $\ell \to \infty$ for sufficiently small $\{\epsilon_\ell\}$, under which $\boldsymbol{\phi}^*_{p,q_\infty}(x) \equiv 0$ and $\boldsymbol{T}^*_\infty$ reduces to the identity map. Now that we know the optimal perturbation direction, we still have to compute it in practice according to (3.13). This can be done by drawing a set of particles from the initial distribution q_0 and then iteratively update the particles with an approximation of (3.14).

3.2.5 SVGD Algorithm

For the sake of consistency in notation, we redefine the variables used in this section to match the ones used in the section on variational inference (§3.1). Formally, let $\mathbf{w}^{(i)} \in \mathbb{R}^m$ be the i^{th} particle, m the number of parameters (mean and standard deviations for each weight and bias) in the network, n the number of particles and $\mathbf{W} = \{\mathbf{w}^{(i)}\}_{i=1}^{i=n}$ the set of all particles. From (3.13), the optimal perturbation direction for particle i is

$$\hat{\boldsymbol{\phi}}^\star(\mathbf{w}^{(i)}) = \mathbf{K}^\top \nabla_{\mathbf{w}^{(i)}} \log \Pr\left(\mathbf{w}^{(i)} \mid \mathcal{D}\right) + \nabla_{\mathbf{w}^{(i)}} \mathbf{K}^\top \tag{3.15}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the gram matrix of the particles. The algorithm then consists of iteratively updating $\boldsymbol{\phi}$ and the position of the particles similarly to standard gradient descent

$$\mathbf{w}^{(i)} \leftarrow \mathbf{w}^{(i)} + \epsilon \hat{\boldsymbol{\phi}}^{\star}(\mathbf{w}^{(i)}) \tag{3.16}$$

Chapter 4

Function-Space View

Recent work has attempted to directly approximate the predictive posterior distribution rather than the posterior distribution over model parameters. This seems more intuitive as the latter does not have a useful meaning that we can directly link to explicable uncertainty in function space. In this section, we present in detail the mechanics and intuition of the functional models used in our experiments, namely FVI and FSVGD. In doing so, we present methods for estimating the score function of an implicit distribution which is needed for both models.

4.1 Stochastic Processes

Consider fitting a function f to our data $\mathcal{D}$. We place a *prior* distribution over f that we denote $p(f)$. Similarly, we place a *posterior* distribution $q_{\boldsymbol{\theta}}(f)$ that depends on parameters $\boldsymbol{\theta}$. Then p and q are distributions over functions and not parameters, i.e. stochastic processes. This means that cannot easily compute the log density $\log q_{\boldsymbol{\theta}}(\cdot)$ of the posterior as we did in weight space because there isn't a closed form expression for it. Therefore trying to maximize an ELBO in function space would not work since the log posterior and log prior terms cannot be evaluated. The rest of this section is concerned with ways to derive a function-space ELBO from theoretical results in stochastic processes. Let us begin by defining some fundamental notions.

Definition 4.1 (Index Set)**.** A set whose members index members of another set.

We can use the concept of index set to informally define a stochastic process.

Definition 4.2 (Stochastic process)**.** A collection of random variables ordered by an index set.

Note that the index set of a stochastic process is always infinite. Formally, a stochastic process F is defined as a collection of random variables, on a probability space $(\Omega, \mathcal{F}, P)$. The random variables, indexed by some set $\mathcal{X}$, all take values in the same space $\mathcal{Y}$. In other words, given a probability space (Ω, Σ, P), a stochastic process can be simply written as $\{F(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$. For any finite index set $\mathbf{x}_{1:n} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we can define the finite-dimensional marginal joint distribution over function values $\{F(\mathbf{x}_1), \cdots, F(\mathbf{x}_n)\}$. For example, Gaussian Processes have marginal distributions as multivariate Gaussians.

4.1.1 A Discrepancy for Stochastic Processes

In order to allow for a variational treatment, we need to minimize the KL divergence between stochastic processes q and p. This does not have a convenient form like (3.2) as explained in Matthews et al. (2015) because there is no infinite-dimensional Lebesgue measure (Eldredge, 2016). In other words, a stochastic process can be seen as an infinite-dimensional random vector which is unpractical. According to the Kolmogorov Extension Theorem (Øksendal, 2003), we can characterize a stochastic process by its finite dimensional marginals, i.e. the distribution when evaluating the random function in a finite number of points which we call *measurement set* in this case. Formally, we define a finite measurement set $\mathbf{X}$ and make use of the main result from Sun et al. (2019) shown in Theorem 4.1 which equates the function space KL divergence to the supremum of marginal KL divergences over all finite measurement sets.

Theorem 4.1. *For two stochastic processes P and Q,*

$$\mathrm{KL}(P, Q) = \sup_{n \in \mathbb{N},\ \mathbf{X} \in \mathcal{X}^n} \mathrm{KL}\left(P_{\mathbf{X}}, Q_{\mathbf{X}}\right) \tag{4.1}$$

where $P_{\mathbf{X}} := P(f(\mathbf{X}))$ is the density P of function f evaluated at $\mathbf{X}$.

Note that minimizing the functional KL divergence with respect to the posterior parameters $\boldsymbol{\theta}$ is a nonlinear dual optimization problem and can quickly get intractable

$$\min_{\boldsymbol{\theta}} \sup_{n\in\mathbb{N},\ \mathbf{X}\in\mathcal{X}^n} \mathrm{KL}\left(p_{\mathbf{X}}, q_{\mathbf{X}}\right) \tag{4.2}$$

One solution would involve solving it in an adversarial manner for a fixed number of samples n. That is, frame it as a non-cooperative zero-sum game where one player chooses the stochastic network parameters θ and the other the measurement points $\mathbf{X}$

$$\min_{\boldsymbol{\theta}} \sup_{\mathbf{X}\in\mathcal{X}^n} \mathrm{KL}\left(p_{\mathbf{X}}, q_{\mathbf{X}}\right) \tag{4.3}$$

This method does not generalize in practice according to Sun et al. (2019) which instead resort to a much simpler approximation scheme: they take the expectation instead of the minimum of the functional KL divergence.

4.1.2 Functional ELBO

The optimization problem to solve at training time is similar to the weight space ELBO in (3.1), except that the distributions are over functions rather than weights. Following from Theorem 4.1, we can define the following optimization objective

$$\begin{aligned} L(q) &= \mathop{\mathrm{E}}_{f\sim q} \log \Pr(\mathcal{D} \mid f) - \sup_{n\in\mathbb{N},\mathbf{X}\in\mathcal{X}^n} \mathrm{KL}\left(q_{\mathbf{X}}, p_{\mathbf{X}}\right) \\ &= \inf_{n\in\mathbb{N},\mathbf{X}\in\mathcal{X}^n} \sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}} \mathop{\mathrm{E}}_{f\sim q} \log \Pr\left(y_i \mid f\left(\mathbf{x}_i\right)\right) - \mathrm{KL}\left(q_{\mathbf{X}}, p_{\mathbf{X}}\right) \\ &:= \inf_{n\in\mathbb{N},\mathbf{X}\in\mathcal{X}^n} L_{\mathbf{X}}(q) \end{aligned} \tag{4.4}$$

For the restriction to a fixed number of elements in the measurement set we define

$$L_n(q) := \inf_{\mathbf{X}\in\mathcal{X}^n} L_{\mathbf{X}}(q) \tag{4.5}$$

The objective function in (4.4) is called the functional evidence lower bound (fELBO). Sun et al. (2019) show that the approximation of the fELBO is in fact a lower bound on the log marginal likelihood if the measurement set is chosen to include all of the training inputs, i.e. $\mathbf{X} = \mathbf{X}^D$ where the training dataset is defined as $\mathcal{D} = (\mathbf{X}^D, \mathbf{Y}^D)$. Consider a distribution c that is uniform in the space that we are interested in making predictions. For shortness, denote by $q_{\mathbf{X}}$ the density q of function f evaluated at finite index points $\mathbf{X}$, idem for $p_{\mathbf{X}}$. One difficulty when dealing with computing the functional KL divergence is that we have a complicated min-max optimization problem to solve as seen in (4.3). For this reason, Sun et al. (2019) propose to choose c as a uniform distribution in the function space, i.e. choose a measurement set at random with each point in the region where we are interested in making predictions having the same probability. This is equivalent to replacing the objective in (4.5) with

$$\underset{\mathbf{X}^D \sim \mathcal{D}}{\mathrm{E}} \; \underset{\mathbf{X}^M \sim c}{\mathrm{E}} L_{\mathbf{X}}(q) \tag{4.6}$$

where $\mathbf{X}^D \sim \mathcal{D}$ denotes sampling training points uniformly from $\mathcal{D}$ and $\mathbf{X}$ is the concatenation of $\mathbf{X}^M$ and $\mathbf{X}^D$.

4.2 Gradient Estimation

A major drawback of the function-space view is that we can no longer evaluate the density of the posterior leading to a dead-end in terms of computing the KL divergence. Fortunately, solely the gradient of the KL divergence is required for its minimization. This leads to the investigation of gradient estimation techniques. The gradient of the log density of a distribution is referred to as the *score* function. This section focuses on methods centered around estimating this quantity.

4.2.1 Score Matching

The idea of estimating the gradient of the log density with respect to input variables (*i.e.* the score function) from samples appeared initially in Hyvärinen (2005).

Assume we observe a random vector $\mathbf{x} \in \mathbb{R}^n$ which has a probability density function denoted by $f_{\mathbf{x}}(.)$. We have a parameterized density model $f(\cdot; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is an m-dimensional vector of parameters. We want to estimate the parameter $\boldsymbol{\theta}$ from $\mathbf{x}$, i.e. we want to approximate $f_{\mathbf{x}}(.)$ by $f(\cdot; \hat{\boldsymbol{\theta}})$ for the estimated parameter value $\hat{\boldsymbol{\theta}}$. The problem is that we only are able to compute the PDF given by the model up to a multiplicative constant $Z(\boldsymbol{\theta})$ called the *partition* function which often turns out being intractable:

$$f(\boldsymbol{\xi};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} h(\boldsymbol{\xi};\boldsymbol{\theta}) \tag{4.7}$$

where $h(\cdot;\boldsymbol{\theta})$ is an unnormalized density parameterized by $\boldsymbol{\theta}$. The score can then be written as

$$g(\boldsymbol{\xi};\boldsymbol{\theta}) := \nabla_{\boldsymbol{\xi}} \log f(\boldsymbol{\xi};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\xi}} \log q(\boldsymbol{\xi};\boldsymbol{\theta}) \tag{4.8}$$

since the partition function is independent of the input $\boldsymbol{\xi}$. We can therefore estimate the score function by minimizing the expected squared distance between the *model* score function $g(\boldsymbol{\xi};\theta)$ and the *data* score function $g_{\mathbf{x}}(\boldsymbol{\xi}) := \nabla_{\boldsymbol{\xi}} p_{\mathbf{x}}(\boldsymbol{\xi})$ given by

$$L(\boldsymbol{\theta}) = \int_{\mathbb{R}^n} p_{\mathbf{x}}(\boldsymbol{\xi}) \left\| g(\boldsymbol{\xi};\boldsymbol{\theta}) - g_{\mathbf{x}}(\boldsymbol{\xi}) \right\|^2 \mathrm{d}\boldsymbol{\xi} \tag{4.9}$$

However $p_{\mathbf{x}}$ is unknown. By applying integration by parts, Hyvärinen (2005) shows that there is a new objective J such that $J(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + C$ where C is does not depend on $\boldsymbol{\theta}$. That means that the minimization of (4.9) is equivalent to minimizing

$$J(\boldsymbol{\theta}) = \mathop{\mathrm{E}}_{\boldsymbol{\xi}\sim p_{\mathbf{x}}} \left[\operatorname{tr} \nabla_{\boldsymbol{\xi}} g(\boldsymbol{\xi};\boldsymbol{\theta}) + \frac{1}{2} \left\| g(\boldsymbol{\xi};\boldsymbol{\theta}) \right\|_2^2 \right] \tag{4.10}$$

which involves the computation of the diagonal elements of the Hessian of the model log-density and can be a bottleneck in high dimensional settings like neural networks.

4.2.2 Sliced Score Matching

As we saw earlier, score matching is suitable for learning unnormalized statistical models. However, it requires the computation of the diagonal of the Hessian of the model's log-density, which makes it difficult to use with deep neural networks. Song et al. (2020) propose a variant of score matching that can scale to deep neural networks and high dimensional data. The central difference is that instead of directly matching the high-dimensional scores, they match their projections along random directions. This is more efficient since it only requires the computation of Hessian-vector products. This framework can be adapted for score estimation which is useful in our setting.

Consider a multivariate distribution $p_{\mathbf{v}}$ and N data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. We draw M projection vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_M\}$ independently for each data point from $p_{\mathbf{v}}$. The j^{th} projection vector for the i^{th} data point is denoted $\mathbf{v}_{ij}$. We then use the following unbiased estimator for $J(\boldsymbol{\theta})$:

$$\hat{J}(\boldsymbol{\theta}) = \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M}\mathbf{v}_{ij}^{\top}\nabla_{\mathbf{x}}g(\mathbf{x}_i;\boldsymbol{\theta})\,\mathbf{v}_{ij} + \frac{1}{2}\left(\mathbf{v}_{ij}^{\top}g(\mathbf{x}_i;\boldsymbol{\theta})\right)^2 \tag{4.11}$$

Song et al. (2019) claim that this method has a computational advantage of a factor D per backward pass where D represents the dimension of the input data.

4.2.3 Stein Gradient Estimator

Li and Turner (2018) frame the problem of score estimation as a regularized regression problem. Li and Turner (2018) further prove that their *Stein gradient estimator* minimizes the kernelized Stein discrepancy. However this method only provides gradient estimates at sample points. This is particularly inconvenient if the measurement points are chosen at random since their location might lead to instability in estimation.

4.2.4 Spectral Stein Gradient Estimator

Shi et al. (2018) propose a gradient estimator for implicit distributions based on Stein's identity and a spectral decomposition of kernel operators, where the eigenfunctions are approximated by the Nyström method. To approximate the score function, we first expand it in terms of the eigenfunctions of a kernel-based operator. These eigenfunctions are orthogonal with respect to the underlying distribution. By setting the test functions in the Stein's identity to these eigenfunctions, we can take advantage of their orthogonality to obtain a simple solution. The eigenfunctions in the solution are then approximated by the Nyström method (Nyström, 1930). Unlike Li and Turner (2018), this approach allows for a direct out-of-sample extension. Recall that we only have access to i.i.d. samples $\{\mathbf{x}^1, \ldots, \mathbf{x}^M\}$ from the distribution of which we are trying to estimate the score (i.e. functional posterior or prior in this case). Note that for this section, q is an arbitrary probability measure and does not relate to the posterior. Since this is the estimator we use in our experiments, we examine its derivation in more detail.

Formally, let $\mathbf{x}$ denote a d-dimensional vector in $\mathbb{R}^d$. Consider an implicit distribution $q(\mathbf{x})$ supported on $\mathcal{X} \subset \mathbb{R}^d$, from which we observe M i.i.d. samples $\{\mathbf{x}^1, \ldots, \mathbf{x}^M\}$. We denote the target score function by $\mathbf{g} : \mathcal{X} \to \mathbb{R}^d$ defined as $\mathbf{g}(\mathbf{x}) = \nabla_{\mathbf{x}} \log q(\mathbf{x})$. The i^{th} component of the gradient is $g_i(\mathbf{x}) = \nabla_{x_i} \log q(\mathbf{x})$. We assume $g_1, \ldots, g_d \in L^2(\mathcal{X}, q)$.

Nyström Method

We wish to find the eigenfunctions $\{\psi_j\}_{j\geq 1}$ of the covariance kernel $k(x, y)$ with respect to the probability measure q, that is, we want to solve the following for ψ:

$$\int k(x, y)\psi(y)q(y)\mathrm{d}y = \mu\psi(x) \tag{4.12}$$

By approximating the left side of (4.12) with its Monte Carlo estimate using M independent samples $\left\{\mathbf{x}^1, \ldots, \mathbf{x}^M\right\}$ from q, we obtain the following eigenvalue problem

$$\frac{1}{M}\mathbf{K}\boldsymbol{\psi} \approx \mu\boldsymbol{\psi} \tag{4.13}$$

where $K_{ij} = k(\mathbf{x}^i, \mathbf{x}^j)$ are the elements of the gram matrix of the sample points and $\boldsymbol{\psi} = \left[\psi\left(\mathbf{x}^1\right), \ldots, \psi\left(\mathbf{x}^M\right)\right]^\top$ is the evaluation of ψ at the sample points. This is an eigenvalue problem for $\mathbf{K}$. We compute the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_J$ with the J largest eigenvalues $\lambda_1 \geq \cdots \geq \lambda_J$ for $\mathbf{K}$. Now we have the solutions of (4.13) by comparing against $\mathbf{K}\mathbf{u}_j = \lambda_j\mathbf{u}_j$:

$$\psi_j\left(\mathbf{x}^m\right) \approx \sqrt{M}u_{jm} \tag{4.14}$$

and $\mu_j \approx \dfrac{\lambda_j}{M}$ for all $m \in [M]$. Plugging these solutions back into (4.12), we get the Nyström formula for approximating the value of the j^{th} eigenfunction at any point $\mathbf{x}$:

$$\psi_j(\mathbf{x}) \approx \hat{\psi}_j(\mathbf{x}) = \frac{\sqrt{M}}{\lambda_j}\sum_{m=1}^{M} u_{jm}k\left(\mathbf{x}, \mathbf{x}^m\right) \tag{4.15}$$

Spectral Expansion

Because the eigenfunctions form an orthonormal basis of $L^2(\mathcal{X}, q)$, which denotes the space of all square-integrable functions with respect to q, we can expand the score into an infinite spectral series as follows:

$$g_i(x) = \sum_{j=1}^{\infty} \beta_{ij} \psi_j(x) \tag{4.16}$$

For the computation of the weights, we state a proposition deriving from Theorem 3.1.

Proposition 4.1. *If $k(\cdot,\cdot)$ has continuous second order partial derivatives, and both $k(\mathbf{x},\cdot)$ and $k(\cdot,\mathbf{x})$ are in the Stein class of q, the following set of equations hold true:*

$$\mathop{\mathrm{E}}_{q}\left[\psi_j(\mathbf{x})g(\mathbf{x}) + \nabla_{\mathbf{x}}\psi_j(\mathbf{x})\right] = \mathbf{0} \tag{4.17}$$

for any positive integer j.

Li and Turner (2018) explain that for an RBF kernel k (which is the one used in all our experiments), and for any fixed $\mathbf{x}$, $k(\mathbf{x},\cdot)$ and $k(\cdot,\mathbf{x})$ are in the Stein class of continuous differentiable densities supported on $\mathbb{R}^d$.

Substituting (4.16) into (4.17) and using the orthonormality of $\{\psi_j\}$, Shi et al. (2018) show that

$$\beta_{ij} = -\mathop{\mathrm{E}}_{q} \nabla_{x_i}\psi_j(\mathbf{x}) \tag{4.18}$$

To estimate β_{ij}, we need an approximation of $\nabla_{x_i}\psi_j(\mathbf{x})$. The key observation is that derivatives can be taken with respect to both sides of (4.12):

$$\begin{aligned}\mu_j \nabla_{x_i}\psi_j(\mathbf{x}) &= \nabla_{x_i}\int k(\mathbf{x},\mathbf{y})\psi_j(\mathbf{y})q(\mathbf{y})d\mathbf{y} \\ &= \int \nabla_{x_i}k(\mathbf{x},\mathbf{y})\psi_j(\mathbf{y})q(\mathbf{y})d\mathbf{y}\end{aligned} \tag{4.19}$$

Using Monte-Carlo sampling with (4.19), we have an estimate of $\nabla_{x_i}\psi_j(\mathbf{x})$:

$$\hat{\nabla}_{x_i}\psi_j(\mathbf{x}) \approx \frac{1}{\mu_j M}\sum_{m=1}^{M} \nabla_{x_i} k\left(\mathbf{x},\mathbf{x}^m\right)\psi_j\left(\mathbf{x}^m\right) \tag{4.20}$$

Substituting (4.14) into (4.20) and comparing with (4.15), we can show

$$\hat{\nabla}_{x_i}\psi_j(\mathbf{x}) \approx \nabla_{x_i}\hat{\psi}_j(\mathbf{x}) \tag{4.21}$$

Now truncating the series expansion to the first J terms and plugging in the Nyström approximations of $\{\psi_j\}_{j=1}^{J}$, we get our estimator:

$$\hat{g}_i(\mathbf{x}) = \sum_{j=1}^{J} \hat{\beta}_{ij}\hat{\psi}_j(\mathbf{x}) \tag{4.22}$$

$$\hat{\beta}_{ij} = -\frac{1}{M}\sum_{m=1}^{M} \nabla_{x_i}\hat{\psi}_j\left(\mathbf{x}^m\right) \tag{4.23}$$

where $\hat{\psi}_j$ is the Nyström approximation of ψ_j as in (4.15). The function $\hat{g}(\cdot)$ is called the spectral Stein gradient estimator (SSGE).

4.3 Function-Space Variational Inference

Now that we have a way to compute and evaluate the score function of implicit distributions at arbitrary measurement points, we define the training optimization problem deriving from (4.6).

$$\max_{q\in\mathcal{Q}} \mathop{\mathrm{E}}_{\mathbf{X}^D\sim\mathcal{D}} \mathop{\mathrm{E}}_{\mathbf{X}^M\sim c} L_{\mathbf{X}}(q) \tag{4.24}$$

where $\mathbf{X} := [\mathbf{X}^D, \mathbf{X}^M]$.

4.3.1 Estimating the functional KL Divergence

An important part of functional variational inference involves calculating the functional KL divergence between functional prior p and functional variational posterior q. Let $\mathbf{X}$ be a set of index points. The functional KL divergence evaluated at $\mathbf{X}$ is

$$\mathrm{KL}(q_{\mathbf{X}}, p_{\mathbf{X}}) = \mathop{\mathrm{E}}_{\boldsymbol{\theta}\sim q} \log q_{\mathbf{X}} + \mathop{\mathrm{E}}_{\xi}\left[f(\mathbf{X};\theta,\xi)\left(\log q_{\mathbf{X}} - \log p_{\mathbf{X}}\right)\right] \tag{4.25}$$

Therefore differentiating it with respect to model parameters, we obtain a simplifiable expression (omitting distribution arguments for simplicity)

$$\nabla_{\boldsymbol{\theta}} \mathrm{KL}(q_{\mathbf{X}}, p_{\mathbf{X}}) = \mathop{\mathrm{E}}_{\boldsymbol{\theta}\sim q} [\nabla_{\boldsymbol{\theta}} \log q_{\mathbf{X}}] + \mathop{\mathrm{E}}_{\xi} [\nabla_{\boldsymbol{\theta}} f(\mathbf{X}) \, (\nabla_f \log q_{\mathbf{X}} - \nabla_f \log p_{\mathbf{X}})] \tag{4.26}$$

$$= \nabla_{\boldsymbol{\theta}} f_{\mathbf{X}} (\nabla_f \log q_{\mathbf{X}} - \nabla_f \log p_{\mathbf{X}}) \tag{4.27}$$

where $\nabla_{\boldsymbol{\theta}} f_{\mathbf{X}}$ is the jacobian of the output function f with respect to model parameters $\boldsymbol{\theta}$ evaluated and computed at index points x. The expectation on the second term is taken over the number of functions considered at each iteration. For instance, if we choose to have 10 stochastic networks performing a forward pass over the same batch of data, the expectation is taken over the 10 different stochastic outputs. The first term in (4.27) can be understood as the negative model entropy which we wish to minimize. The second term in (4.27) is the gradient of the cross-entropy of the posterior and the prior which we would like to minimize too. As mentioned earlier, these are now stochastic processes and don't have a closed for density as in the weight-space case. However we can evaluate their score function at finite measurement sets using gradient estimators.

Remark. We chose to use exclusively SSGE for the functional KL gradient estimation in this thesis.

4.3.2 FVI Algorithm

In this section we present the full algorithm of functional variational inference for Bayesian neural networks displayed in Algorithm 1. Here, $g_{\mathbf{w}}$ is a neural network parameterised by vectorized weights and biases $\mathbf{w}$. The data-generating process is denoted $\nu : \mathcal{X} \to \mathcal{Y}$ and is parameterised by a vector $\boldsymbol{\omega}$.

Algorithm 1: Functional Variational Inference (FVI)

```
Input: Dataset $\mathcal{D}$, variational posterior $q_{\boldsymbol{\theta}}$, prior $p$, sampling distribution $c$,
       a neural network $g_{\mathbf{w}}$, a data-generating process $\nu(\cdot)$
for $N$ training iterations do
    Sample $\mathbf{X}^M \sim c$ and $(\mathbf{X}^D, \mathbf{Y}^D) \subset \mathcal{D}$ and stack $\mathbf{X} = [\mathbf{X}^M, \mathbf{X}^D]$;
    Sample $k$ configurations $\boldsymbol{\omega}_i$;
    Sample $k$ values for each element in the measurement set $\mathbf{X}^M$ from the
      data-generating process $\mathbf{Y}_i^M \sim \nu_{\boldsymbol{\omega}_i}(\mathbf{X}^M)$;
    Sample $k$ neural network parameters from variational posterior $\mathbf{w}_i \sim q_{\boldsymbol{\theta}}$;
    Make forward passes through NN $\mathbf{Y}_i = g_{\mathbf{w}_i}(\mathbf{X})$;
    Compute log likelihood gradients $\Delta_1 = \frac{1}{k}\sum_{i=1}^{k} \nabla_{\boldsymbol{\theta}} \log \Pr(\mathbf{Y}^D \mid \mathbf{Y}_i^D, \mathbf{X}^D)$;
    Estimate KL gradients $\Delta_2 = \nabla_{\boldsymbol{\theta}} \mathrm{KL}(q_{\mathbf{X}}, p_{\mathbf{X}})$ using SSGE;
end
```

4.4 Function-Space SVGD

It turns out we can also use particle optimization techniques for function space inference. A nice property of the function space inference is that it does not suffer from the *over-parameterization* problem. However, as we saw earlier (§4.3) it is hard to implement, as the function space is infinite-dimensional, and the prior on it is implicitly defined. Wang et al. (2019) present a simple yet effective solution to this problem inspired by SVGD (Liu and Wang, 2016).

4.4.1 Exact version

Assume that $\mathcal{X}$ is a finite set and that $\nabla_f \log p(f(\mathbf{x}))$ is available for any $\mathbf{x}$. Consider a set of particles represented in function space $f_1(\mathcal{X}), \ldots, f_n(\mathcal{X})$ and are updated iteratively,

$$f(\mathcal{X}) \leftarrow f(\mathcal{X}) - \epsilon\, \mathbf{v}\left(f(\mathcal{X})\right) \tag{4.28}$$

Note that this update rule promotes inspection of the whole dataset at each iteration.

4.4.2 Sampling-based version

To make the computational efficiency of each iteration independent of the size of the dataset, we can use a finite set of samples $\mathbf{X} \subset \mathcal{X}$ drawn from an arbitrary distribution c similarly to function-space variational inference covered in the previous section. Additionally, we approximate function-space particles in weight-space to obtain a new update rule

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}\right) \mathbf{v}\left(f\left(\mathbf{X}\right)\right) \tag{4.29}$$

where $\nabla_{\boldsymbol{\theta}} f(\mathbf{X})$ is the jacobian of the output function with respect to model parameters evaluated at the sampled index points.

4.4.3 Choice of gradient flow

The choice of $\mathbf{v}$ is analogous to the weight-space version of SVGD. It is again composed of a log posterior gradient term and a repulsive force term:

$$\mathbf{v}(f(\mathbf{x}; \theta_i)) = -\frac{1}{n}\left[\sum_{j=1}^{n} \mathbf{K}_{ij} \nabla_{\theta_j} \log \Pr\left(\theta_j \mid \mathbf{x}\right) + \nabla_{\theta_j} \mathbf{K}_{ij}\right] \tag{4.30}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the gram matrix of the particles. Note that the computation of the gradient flow requires two kernel evaluations. The first one dictates how particles will influence each other depending on their distance in weight space. The second one acts as a repulsive force in between particles and can be expressed either in the weight or the function space.

4.4.4 FSVGD Algorithm

In this section, we include a pseudo code for functional SVGD. The notation mirrors the functional variational inference section (§4.3.2).

Algorithm 2: Functional Stein Variational Gradient Descent (FSVGD)

```
Input: function-space prior $p$, training set $\mathcal{D}$, a continuous distribution $c$
       supported on $\mathcal{X}$ and a set of initial particles $\{\mathbf{w}_0^1, \ldots, \mathbf{w}_0^n\}$
for $N$ training iterations do
    Sample $\mathbf{X}^M \sim c$ and $(\mathbf{X}^D, \mathbf{Y}^D) \subset \mathcal{D}$ and stack $\mathbf{X} = [\mathbf{X}^M, \mathbf{X}^D]$;
    Sample $k$ values for each element in the measurement set $\mathbf{X}^M$ from the
      data-generating process $\mathbf{Y}_i^M \sim \nu_{\boldsymbol{\omega}_i}(\mathbf{X}^M)$;
    Compute the gradient flow in function space $\mathbf{v}\left(f(\mathbf{X})\right)$ according to
      (4.30);
    Update particles according to (4.29);
end
```

Chapter 5

Experiments

In this section we present the results of our experiments and give more insight about the details of the experimental procedure. The models are evaluated after 10,000 training iterations and a batch size of 8. Note that the non-functional Bayesian models (VI, SVGD and PACOH) cannot benefit from the prior transfer. Hyperparameter search is performed using a combination of the distributed python libraries Ray-Tune (Liaw et al., 2018) and HyperOpt (Bergstra et al., 2015). Most hyperparameters are sampled log-uniformly 100 times for each model per dataset. Also the box-plots displayed show the aggregated results over 10 runs with different global seed, or equivalently, 10 separate tasks. In all the experiments, the posterior over model parameters is represented as a stochastic neural network with independent Gaussian distributions over the weights. The experiments are displayed in increasing order of dimensional complexity. The models were trained using the same data for all environments and the number of training points varied according to the task's complexity.

Metrics

As is standard in Bayesian deep learning, we make use of the negative log likelihood (NLL) and root mean square error (RMSE) metrics. For predictive mean $\hat{\mu}$, standard deviation $\hat{\sigma}$ and ground truth y, we have

$$\mathrm{NLL}(y; \hat{\mu}, \hat{\sigma}) = -\log[\mathcal{N}(y; \hat{\mu}, \hat{\sigma})] \tag{5.1}$$

where $\mathcal{N}(x; \mu\sigma)$ is the Gaussian density function with mean μ and standard deviation σ evaluated at x. The other metric is the root mean square error,

$$\text{RMSE}(y, \hat{\mu}) = \|y - \hat{\mu}\|_2^2 \tag{5.2}$$

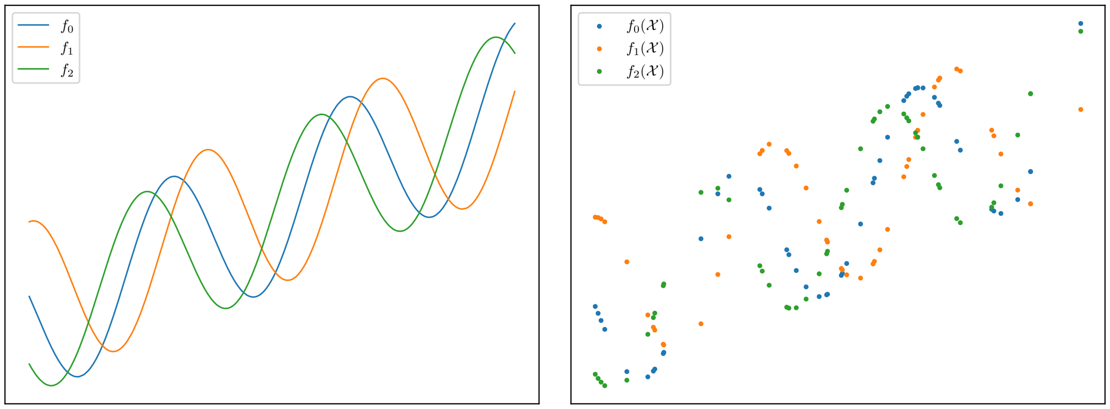
considering one, single-dimensional data point for the sake of simplicity. In the multivariate case, the normal distribution is replaced by a multivariate diagonal Gaussian.

5.1 Sinusoids

A more flexible environment can be generated by sampling index points and evaluating a sinusoid-like function at those points. Contrarily to the Snelson dataset, the sinusoid environment can be seen as a data-generating process in the sense that the amount of data sampled can be infinite. The sampling process goes as follows. The amplitude A, slope m, period T, horizontal shift x_0 and vertical shift y_0 are randomly generated to obtain the function

$$y_0 + mx + A\sin\left[\frac{2\pi}{T}(x - x_0)\right] \tag{5.3}$$

Details about the exact values are displayed in the appendix. Samples from the



(a) Functions (b) Evaluations

Figure 5.1: Samples from the sinusoids environment

sinusoids are displayed in Figure 5.1a. However as shown in Theorem 4.1, it turns out that the KL divergence between two stochastic processes can empirically be computed as the expectation of the KL divergence at n distinct index points called *measurement points* and denoted by $\mathcal{X}$ as shown in Figure 5.1b.

It is clear that for this toy experiment, we would expect function-space models as well as meta-learned priors to perform better than the rest. As is shown in Figure 5.7d, this is clearly the case. Furthermore, both of the models benefiting from function-space priors derived from oracle access data (i.e. FVI and FSVGD), performing better than the PACOH model which only has a finite pool of functions to learn the prior from and more importantly, it needs to explicitly represent the prior as a factorized Gaussian. Additionally, the standard deviation over multiple parameter samples and environment samples is relatively small for all models which confirms the superiority of functional models for this task.

5.2 Densities

We now concern ourselves with regression problems aiming to learn a mapping $f : \mathbb{R}^m \to \mathbb{R}^n$ with $n, m > 1$. Such problems are closer to real world situations as we usually deal with high dimensional data and our models have to be scalable.

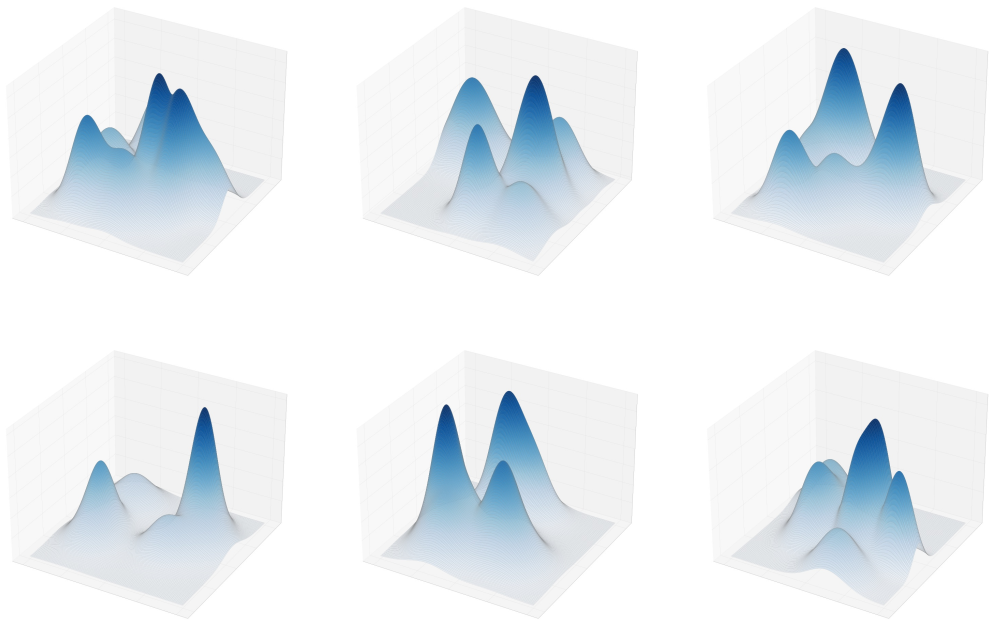


Figure 5.2: Sampled densities configurations

Consider a multivariate vector function $f : \mathbb{R}^m \to \mathbb{R}^n$ of the form

$$f(x) = \begin{bmatrix} \mathcal{N}(x; \mu_1, \Sigma_1) \\ \vdots \\ \mathcal{N}(x; \mu_n, \Sigma_n) \end{bmatrix} \tag{5.4}$$

parameterized by $\mu = [\mu_1, \ldots, \mu_n]$ and $\Sigma = [\Sigma_1, \ldots, \Sigma_n]$. This problem is inspired by the fact that densities are multivariate functions with many desirable properties including summation to 1, inclusion in [0, 1], smoothness, and more. Examples of sampled functions in the case where the input dimension is 2 and the output dimension is 1 are displayed in Figure 5.2. The parameters of the densities generator were sampled from uniform distribution in the intervals [-3, 3] and [0.5, 1] for the means and standard deviations respectively.

This task is similar to the previous one in the sense that it is a toy problem dealing with smooth and differentiable functions with desirable properties for neural networks as stated earlier. However, the function space generated can be arbitrarily complex since there can be more dimensions in the output space than in the input space resulting in a non-bijective transformation. Still, the functional models reveal themselves superior along with the meta-learned model as shown in Figures 5.8a and 5.7a. However the gap between meta-learned weight-space prior and function-space models is reduced compared to the simpler case of the sinusoids.

5.3 MuJoCo

The next few experiments deal with higher dimensional and more complex dynamical systems that often involve the resolution of nonlinear systems and are harder to grasp. MuJoCo (Todorov et al., 2012) is a physics simulator which, combined with OpenAI Gym (Brockman et al., 2016), provides the option to simulate robots and control them accurately.

The task now becomes trying to estimate the next state of the robot given the current state and a random action as shown in Figures 5.3a and 5.3b. This is particularly useful when trying to plan ahead in control. Consider for example a Model Predictive Controller driving a race car. Being able to accurately predict the model dynamics is of outmost importance for the accuracy of the policy.

Concerning the environment, OpenAI Gym offers a variety of options. However,

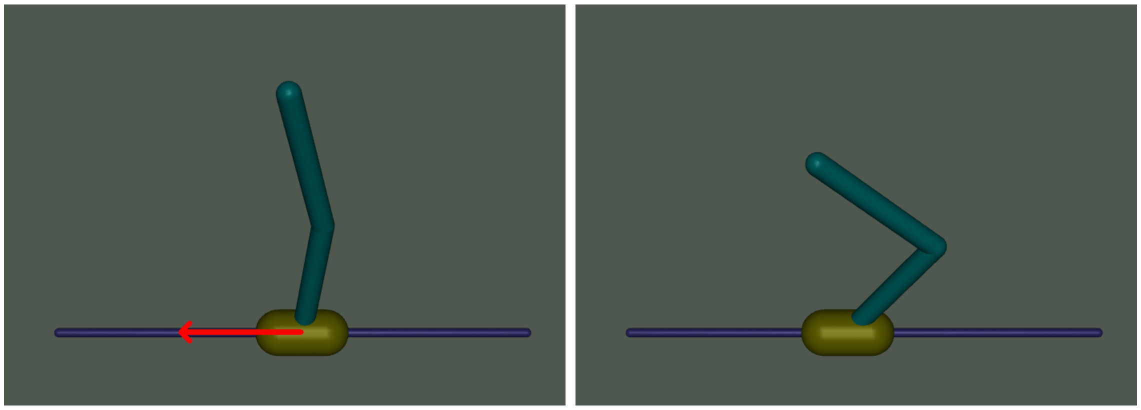

(a) Initial state and action (red arrow) (b) Next state

Figure 5.3: The task of predicting the next state from an initial state and action (red arrow) in the inverted double pendulum environment

in order to avoid infeasible states, too complex (involving contact forces) and high dimensional environments, we chose to restrict our attention to the problems displayed in Table 5.1. Note that because of the fact that our models are given both

Environment	**State Space**	**Action Space**
Inverted Double Pendulum	$\mathbb{R}^8$	$[-1, 1]$
Swimmer	$\mathbb{R}^{11}$	$[-1, 1]^2$
Half Cheetah	$\mathbb{R}^{17}$	$[-1, 1]^6$

Table 5.1: Environments considered and their properties

state and action, the input dimension is larger than the output dimension. For example in the Half Cheetah, the input dimension is $17 + 6 = 23$.

5.3.1 Inverted Double Pendulum

The inverted double pendulum is a classical problem in physics and consists of the resolution of a system of second order ordinary differential equations. In this case, to ensure strict positivity of the parameters being randomized, i.e. length and width of rods and viscosity parameters for the swimmer, we chose a log-normal distribution. Recall that the first and second moments of a random variable $X \sim \mathcal{L}(\mu, \sigma^2)$ are

$$\mathrm{E}[X] = \exp\left(\mu + \frac{\sigma^2}{2}\right), \qquad \mathrm{Var}[X] = \left[\exp\left(\sigma^2\right) - 1\right] \exp\left(2\mu + \sigma^2\right) \tag{5.5}$$

and its support consists of the positive real numbers, that is $x \in (0, +\infty)$.

> **Remark** (MuJoCo Dynamics)**.** Note that changing the length and width of rods does change the dynamics of the simulation since they impact the moment of inertia of the concerned part.

Snapshots of the robot at its initial condition are displayed for 6 different randomness settings (different seeds) in Figure 5.4.

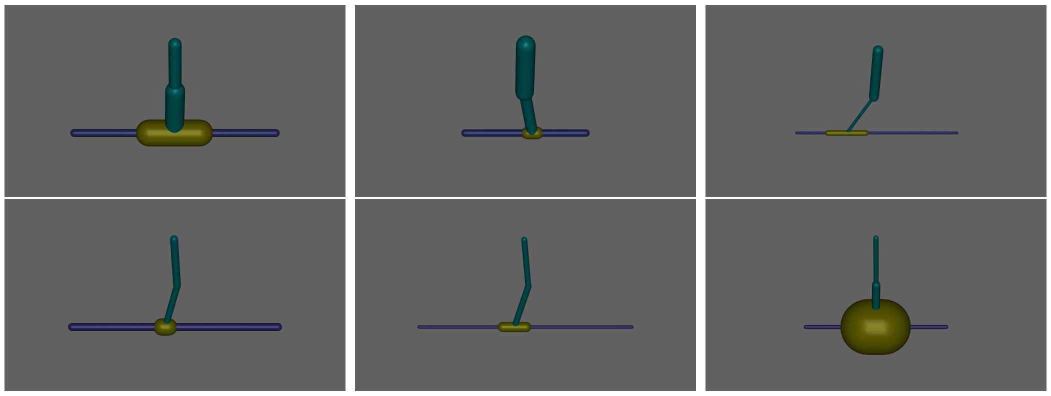


Figure 5.4: Sampled inverted double pendulum configurations

The results for the IDP are also displaying superior performance from functional models as shown in Figures 5.7c and 5.8c. Once again, the meta-learned prior displays slightly worse results than the functional models for both metrics. This is logical even only considering the amount of diverse information the models are given, the functional models have oracle access to the data generating process while the PACOH model only has access to a finite subset of these functions.

5.3.2 Swimmer

As is the case for the inverted double pendulum, a log-normal distribution was used to sample the radius of the rods. The viscosity of the surrounding flow was also randomized in this environment. This directly impacts the grad force on the swimmer. Note that the remark from earlier still applies: dynamics are affected by moment of inertia which is in turn affected by the size of the rods. The viscosity parameter controls the viscous friction between the swimmer and the liquid creating a drag force on the body as explained in Coulom (2002). Snapshots from the randomized

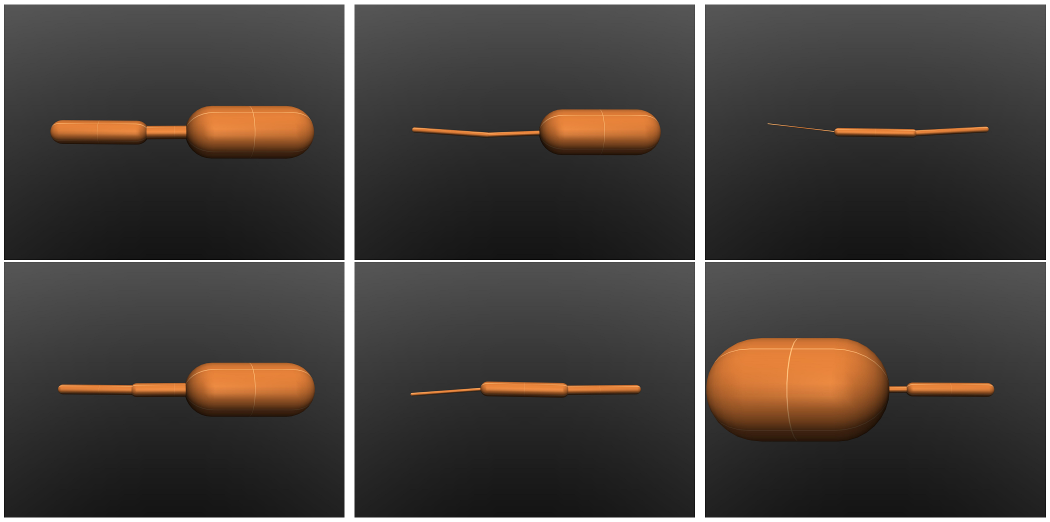


Figure 5.5: Sampled swimmer configurations

environment at its initial state are displayed in Figure 5.5. One can therefore view each of random configuration as a separate parametric function, although they are sharing common characteristics. Recall that the swimmer has a state space of size 11 and an action space of size 2. Hence, it is still relatively simple to grasp for a neural network as will be seen in the results. We will now consider the case of a more challenging environment both in terms of dimensionality of state and action spaces and dynamic complexity.

The swimmer is an environment that includes physics beyond what rigid-body simulations are accustomed to represent. It includes a drag component that depends on a viscosity parameter of the fluid the swimmer is moving in. This presumably complicates the dynamics and is expected to reduce the performance compared to the pendulum nonetheless the dimensionality of the problem not being considerably different. As shown in Figures 5.7e and 5.8e, the functional models are still superior.

5.3.3 Half Cheetah

This environment of OpenAI Gym is the most complex we will look at in this work. It consists of an observation space of size 17 and an action space of size 6, along with two soft contact points. Unlike previous environments, the Half Cheetah has variable joint stiffness in its hinge joints which allows for randomization. Accordingly, sampled configurations are displayed in Figure 5.6 for different seed values. Note that the six actions corresponding to the size of the action space include actuators at both front and back thighs, shins and feet.

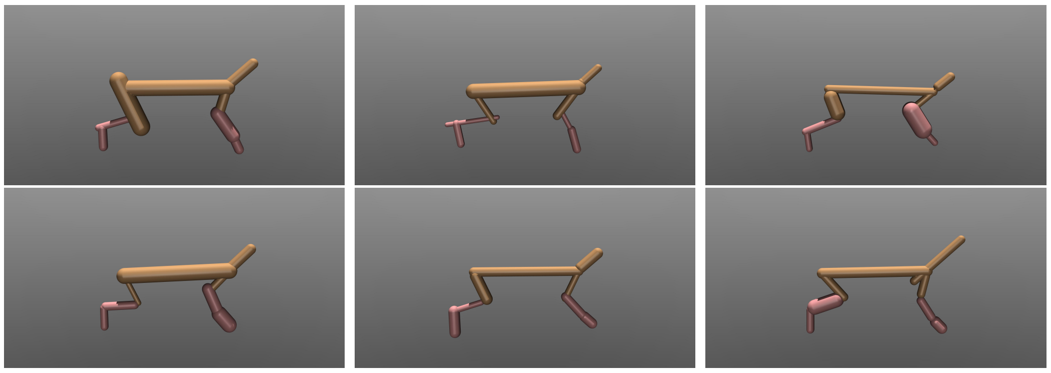


Figure 5.6: Sampled half cheetah configurations

This environment is often times complex for a neural network to learn because of the complexity of the hinge joints with added stiffness. All the models performed poorly compared to the other tasks. Furthermore, the functional models actually score worse in both metrics as shown in Figures 5.7b and 5.8b. This might be due to the gradient estimation being confused by the prior samples in a complex function space hence providing poor estimates to the functional models.

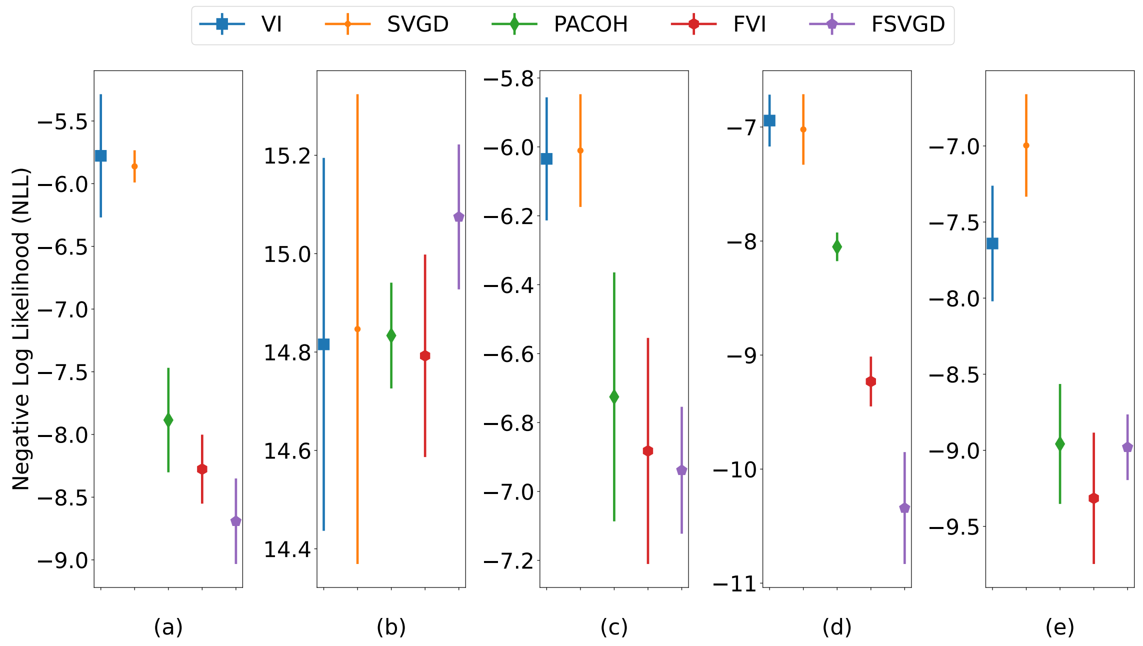


Figure 5.7: NLL error bars for (a) densities, (b) half cheetah, (c) inverted double pendulum, (d) sinusoids and (e) swimmer environments.

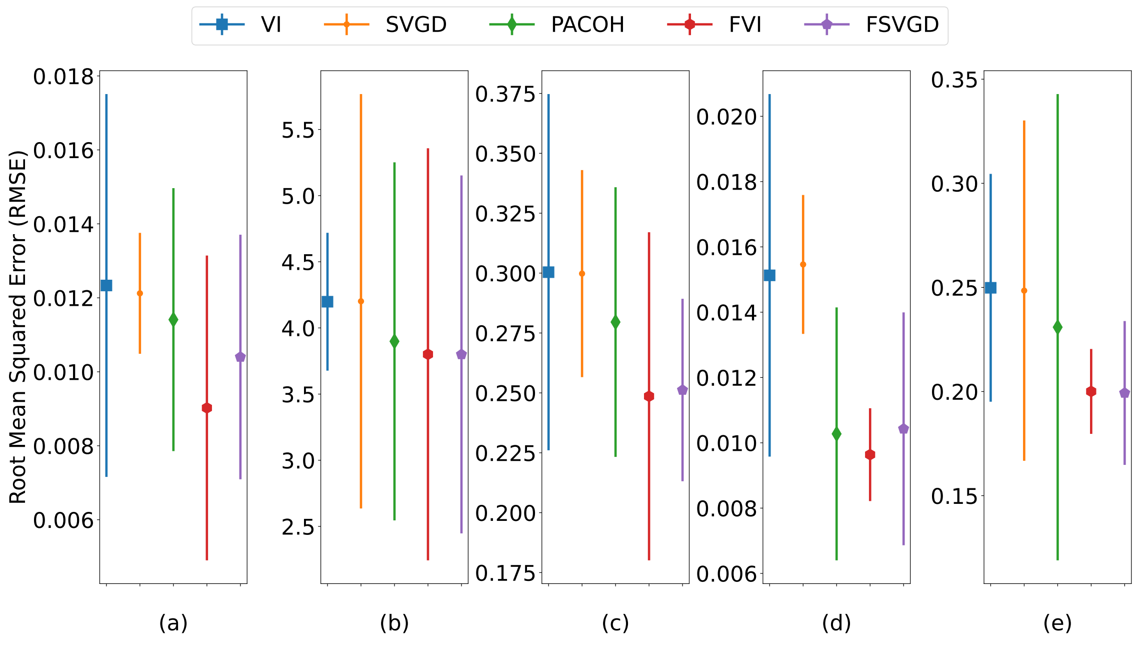


Figure 5.8: RMSE error bars for (a) densities, (b) half cheetah, (c) inverted double pendulum, (d) sinusoids and (e) swimmer environments.

5.4 Summary

From the results in Figures 5.7 and 5.8, we can conclude that function-space models perform better than non-functional ones and that it is therefore possible to use the data-generating process as a prior in Bayesian modeling even for high dimensional and complex parametric spaces like those induced by neural networks.

Chapter 6

Final Remarks

In this final chapter, we first review the main results of the thesis on a high level. Then, we state some domains where our approach could have applicability and present examples from the industry sector. Finally, we discuss future work and possible improvements.

6.1 Conclusion

We demonstrated that the data-generating process can be used as an implicit prior for Bayesian neural networks and particle optimization based variational inference techniques. Through our experiments we observed that the models benefiting from this extra expressiveness in the prior distribution displayed superior results compared to their non-functional counterparts in all cases according to both performance metrics (NLL and RMSE). This is encouraging since working in the function space (i) is more intuitive and (ii) solves the problem of over-parameterization that arises in the parameter space. In summary, we showed that prior information in the form of a probability distribution inherent in the data-generation process can be transferred to an implicit distribution in the parameter space of the model and increase its performance. In the univariate and simple multivariate experiments, we saw a net improvement over the weight-space models and a significant improvement over the benchmark meta learning model, that explicitly learns the prior in weight-space. In more complex environments like the half cheetah, the function-space prior does not seem to offer much of an overhead, due to an increased complexity in the nature of the task.

6.2 Applications

Such a framework can provide a sold basis for application in various domains. For instance, *sim-to-real* (Zhao et al., 2020) is a field that could directly benefit from priors in function space. Its effective bottleneck lies in the *distributional shift* between the training (simulation) and testing (real world) distribution. This means that an agent deployed in the real world will generally be exposed to new scenarios that were not present in the simulations. Usually this issue is met with what is known as *domain randomization* (Muratore et al., 2021), or the introduction of randomness in the environment during training time in hope of bridging the distributional gap.

Instead, using a functional approach, one could use real world data to train and samples from the simulation as a functional prior, as sampling data from the real world can be considerably more expensive. This would lead to lead to a smaller distributional gap and would allow for faster convergence than domain randomization as the training data is taken directly from the target distribution. Another field that might be able to make use of this idea would be reinforcement learning or other sequential decision making algorithms that could leverage the functional prior for more accurate planning.

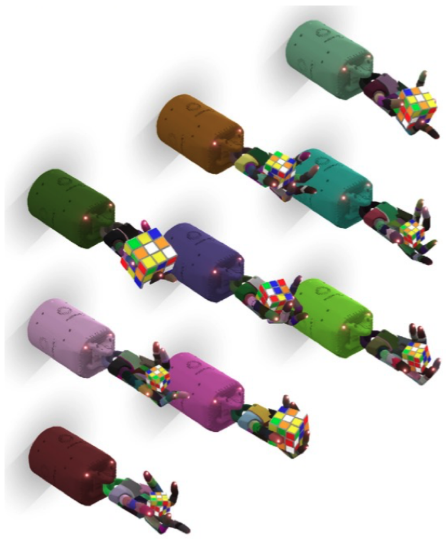


Figure 6.1: Domain randomization for the task of solving Rubik's cube with a robotic hand (OpenAI et al., 2019)

A recent milestone in the fields of domain randomization and simulation to reality (sim-to-real) happened about a year ago when OpenAI et al. (2019) managed to train a robot hand to solve Rubik's cube by transferring knowledge gained from a simulation and training in the real world . This could not have been achieved without a very accurate model of the hand. Domain randomization was used in this case to make the model invariant to parameters that are not relevant to the task of solving the cube (e.g. color, brightness, and more) as shown in Figure 6.1.

6.3 Further Work

An idea that could improve the distribution transfer would be to choose the measurement points, at which the prior is sampled, in an adversarial manner. That is, instead of randomly selecting an index set and sampling the function space prior at these points, one could cast this as a min-max optimization problem. On one side we are trying to maximize the fELBO with respect to the parameters of the model, and on the other we try to minimize it by choosing an index set that would indicate the most interesting and presumably unexplored parts of the output space. The research question in this case would be: how can we make the function space prior invariant (or robust) to the distributional shift between simulation and reality?

The randomization parameters of the data-generating process in this work have been chosen at random. An interesting research direction could be to derive causality relations between the stochasticity (how much different environment configurations differ from each other) of the data-generating process and the model performance.

Bibliography

D. Barber and C. M. Bishop. Ensemble learning in Bayesian neural networks. In *Neural Networks and Machine Learning*, pages 215–237. Springer, 1998.

J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, July 2015. ISSN 1749-4699. doi: 10.1088/1749-4699/8/1/014008. URL https://doi.org/10.1088/1749-4699/8/1/014008. Publisher: IOP Publishing.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Networks. *arXiv:1505.05424 [cs, stat]*, May 2015. URL http://arxiv.org/abs/1505.05424. arXiv: 1505.05424.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, June 2016. URL http://arxiv.org/abs/1606.01540. arXiv: 1606.01540.

M.-H. Chen, Q.-M. Shao, and J. G. Ibrahim. *Monte Carlo Methods in Bayesian Computation*. Springer Series in Statistics. Springer-Verlag, New York, 2000. ISBN 978-0-387-98935-8. doi: 10.1007/978-1-4612-1276-8. URL https://www.springer.com/gp/book/9780387989358.

R. Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD Thesis, Institut National Polytechnique De Grenoble, Grenoble, 2002.

J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield. Large Automatic Learning, Rule Extraction, and Generalization. *Complex Syst.*, 1987.

J. S. Denker and Y. LeCun. Transforming neural-net output levels to probability distributions. In *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, NIPS'90, pages 853–859, San Francisco, CA, USA, Nov. 1990. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-184-0.

N. Eldredge. Analysis and Probability on Infinite-Dimensional Spaces. *arXiv:1607.03591 [math]*, Sept. 2016. URL http://arxiv.org/abs/1607.03591. arXiv: 1607.03591.

Y. Gal. *Uncertainty in Deep Learning*. PhD Thesis, Cambridge University, Sept. 2016.

C. F. Gauss. *Theory Of The Motion Of The Heavenly Bodies Moving About The Sun*. Boston,Little, Brown and company,, 1809.

A. Gelman. Bayesian Model-building By Pure Thought: Some Principles And Examples. *Statistica Sinica*, 6(1):215–232, 1996. ISSN 10170405, 19968507. URL http://www.jstor.org/stable/24306008. Publisher: Institute of Statistical Science, Academia Sinica.

J. Gorham and L. Mackey. Measuring Sample Quality with Stein's Method. *Advances in Neural Information Processing Systems*, 28, 2015. URL https://papers.nips.cc/paper/2015/hash/698d51a19d8a121ce581499d7b701668-Abstract.html.

A. Graves, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011. URL http://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf.

G. E. Hinton and D. van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, COLT '93, pages 5–13, New York, NY, USA, Aug. 1993. Association for Computing Machinery. ISBN 978-0-89791-611-0. doi: 10.1145/168304.168306. URL https://doi.org/10.1145/168304.168306.

B. L. Ho and R. E. Kalman. Effective construction of linear state-variable models from input/output data. In *3rd Allerton Conference*, pages 449–459, Jan. 1965.

A. Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. ISSN 1533-7928. URL http://jmlr.org/papers/v6/hyvarinen05a.html.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. URL http://arxiv.org/abs/1312.6114. arXiv: 1312.6114.

Y. Li and R. E. Turner. Gradient Estimators for Implicit Models. *arXiv:1705.07107 [cs, stat]*, Apr. 2018. URL http://arxiv.org/abs/1705.07107. arXiv: 1705.07107.

R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv:1807.05118 [cs, stat]*, July 2018. URL http://arxiv.org/abs/1807.05118. arXiv: 1807.05118.

Q. Liu and D. Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. *Advances in Neural Information Processing Systems*, 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/b3ba8f1bee1238a2f37603d90b58898d-Abstract.html.

Q. Liu and D. Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. *arXiv:1608.04471 [cs, stat]*, Sept. 2019. URL http://arxiv.org/abs/1608.04471. arXiv: 1608.04471.

C. Louizos and M. Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *arXiv:1703.01961 [cs, stat]*, June 2017. URL http://arxiv.org/abs/1703.01961. arXiv: 1703.01961.

D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, May 1992. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1992.4.3.448. URL https://www.mitpressjournals.org/doi/abs/10.1162/neco.1992.4.3.448.

A. G. d. G. Matthews, J. Hensman, R. E. Turner, and Z. Ghahramani. On Sparse variational methods and the Kullback-Leibler divergence between stochastic processes. *arXiv:1504.07027 [stat]*, Dec. 2015. URL http://arxiv.org/abs/1504.07027. arXiv: 1504.07027.

S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. Monte Carlo Gradient Estimation in Machine Learning. *arXiv:1906.10652 [cs, math, stat]*, Sept. 2020. URL http://arxiv.org/abs/1906.10652. arXiv: 1906.10652.

F. Muratore, C. Eilers, M. Gienger, and J. Peters. Data-efficient Domain Randomization with Bayesian Optimization. *IEEE Robotics and Automation Letters*, 6(2): 911–918, Apr. 2021. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2021.3052391. URL http://arxiv.org/abs/2003.02471. arXiv: 2003.02471.

R. Neal. Bayesian Learning via Stochastic Dynamics. *Advances in Neural Information Processing Systems*, 5, 1992. URL https://papers.nips.cc/paper/1992/hash/f29c21d4897f78948b91f03172341b7b-Abstract.html.

R. M. Neal. *Bayesian Learning for Neural Networks.* Ph.D. Dissertation, University of Toronto, 1995.

E. J. Nyström. Über Die Praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben. *Acta Mathematica*, 54(none):185–204, Jan. 1930. ISSN 0001-5962, 1871-2509. doi: 10.1007/BF02547521. URL https://projecteuclid.org/journals/acta-mathematica/volume-54/issue-none/%c3%9cber-Die-Praktische-Aufl%c3%b6sung-von-Integralgleichungen-mit-Anwendungen-auf-Randwertaufgaben/10.1007/BF02547521.full. Publisher: Institut Mittag-Leffler.

OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv:1910.07113 [cs, stat]*, Oct. 2019. URL http://arxiv.org/abs/1910.07113. arXiv: 1910.07113.

J. Rothfuss, V. Fortuin, M. Josifoski, and A. Krause. PACOH: Bayes-Optimal Meta-Learning with PAC-Guarantees. *arXiv:2002.05551 [cs, stat]*, June 2021. URL http://arxiv.org/abs/2002.05551. arXiv: 2002.05551.

J. Shi, S. Sun, and J. Zhu. A Spectral Approach to Gradient Estimation for Implicit Distributions. *arXiv:1806.02925 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1806.02925. arXiv: 1806.02925.

Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. *arXiv:1905.07088 [cs, stat]*, June 2019. URL http://arxiv.org/abs/1905.07088. arXiv: 1905.07088.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-Based Generative Modeling through Stochastic Differential Equations. *arXiv:2011.13456 [cs, stat]*, Nov. 2020. URL http://arxiv.org/abs/2011.13456. arXiv: 2011.13456.

C. M. Stein. Estimation of the Mean of a Multivariate Normal Distribution. *The Annals of Statistics*, 9(6):1135–1151, Nov. 1981. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1176345632. URL https://projecteuclid.org/journals/annals-of-statistics/volume-9/issue-6/Estimation-of-the-Mean-of-a-Multivariate-Normal-Distribution/10.1214/aos/1176345632.full. Publisher: Institute of Mathematical Statistics.

S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional Variational Bayesian Neural Networks. *arXiv:1903.05779 [cs, stat]*, Mar. 2019. URL http://arxiv.org/abs/1903.05779. arXiv: 1903.05779.

E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012. ISBN 978-1-4673-1737-5. URL http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12.

A. W. v. d. Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998. doi: 10.1017/CBO9780511802256.

Z. Wang, T. Ren, J. Zhu, and B. Zhang. Function Space Particle Optimization for Bayesian Neural Networks. *arXiv:1902.09754 [cs, stat]*, May 2019. URL http://arxiv.org/abs/1902.09754. arXiv: 1902.09754.

Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. *arXiv:1803.04386 [cs, stat]*, Apr. 2018. URL http://arxiv.org/abs/1803.04386. arXiv: 1803.04386.

W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *2020 IEEE Symposium Se-*

ries on Computational Intelligence (SSCI), pages 737–744, Dec. 2020. doi: 10.1109/SSCI47803.2020.9308468.

B. Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Universitext. Springer-Verlag, Berlin Heidelberg, 6 edition, 2003. ISBN 978-3-540-04758-2. doi: 10.1007/978-3-642-14394-6. URL https://www.springer.com/gp/book/9783540047582.

Appendix A

Parameter Randomization in the Data-Generating Process

Sinusoids

The parameters for the sinusoids environment were sampled according to Table A.1.

Parameter		Distribution
Amplitude	A	$\mathcal{U}\left(0.9, 1.1\right)$
Period	T	$\mathcal{U}\left(0.9, 1.1\right)$
Slope	m	$\mathcal{N}\left(1.0, 0.1\right)$
Horizontal shift	x_0	$\mathcal{N}\left(0.0, 0.1\right)$
Vertical shift	y_0	$\mathcal{N}\left(0.0, 0.1\right)$

Table A.1: Parameter randomization for sinusoids

Densities

The parameters for the densities environment were sampled according to Table A.2.

Parameter		Distribution
Mean	μ_i	$\mathcal{U}(-0.5, 0.5)$
Standard deviation	σ_i	$\mathcal{U}(0.25, 0.5)$

Table A.2: Parameter randomization for densities

MuJoCo

OpenAI Gym already provides default values for the environment parameters, i.e. stiffness, viscosity or dimensions. The parameters for the MuJoCo environments were sampled from a log-normal density with the default value of the parameters (from `gym`) as mean and a standard deviation proper to each environment. We used 0.1 as standard deviation for the inverted double pendulum, 0.1 for the swimmer and 0.01 for the half cheetah.