Functional Representation Learning for Uncertainty Quantification and Fast Skill Transfer

Qi Wang

This book was typeset by the author using LaTeX 2_ε.

[TODO] ISBN 000-00-0000-000-0

Functional Representation Learning for Uncertainty Quantification and Fast Skill Transfer

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op woensdag 23 december 2022, te 13:00 uur

door

Qi Wang

geboren te Anhui, China

Promotiecommissie

Promotor:	Prof. dr. M. Wellling	Universiteit van Amsterdam
Co-promotor:	Dr. H.C. Van Hoof	Universiteit van Amsterdam
Overige leden:	Prof. dr. M. de Rijke	Universiteit van Amsterdam
	Prof. dr. C. G. M. Snoek	Universiteit van Amsterdam
	Prof. dr. E. Gavves	Universiteit van Amsterdam
	Prof. dr. A. Plaat	Leiden University
	Dr. S. Magliacane	Universiteit van Amsterdam
	Dr. X. Zhen	United Imaging

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

UNIVERSITEIT VAN AMSTERDAM

The research was supported by the China Scholarship Council under grant #201803170241.

Cover by Someone Else
Printed by Printer Name, City

ISBN: xxx-xx-xxxx-xxx-x

SUMMARY

This thesis aims at *Functional Representation Learning for Uncertainty Quantification & Fast Skill Transfer*. Real-world scenarios are posing increasing practical demands for deep learning models. Two particular considerations are *uncertainty quantification* and *fast skill transfer*. The first consideration has the potential to address risk-sensitive decision-making or reduce sample complexity in query problems. The second consideration is to avoid learning from scratch and increase the adaptive capability of deep learning models.

My thesis is that learning *functional representation* offers a tractable scheme to accomplish the above missions. Unlike representing a single data point with variational autoencoders (VAEs) (Kingma and Welling, 2013), this thesis will focus on representing a set of data points from a function. Throughout this thesis, *the summary of context data points* $\mathcal{D}^C = \{(x_i, y_i)\}_{i=1}^n$ *to represent an underlying function* $y = f(x)$ will be referred to as the *functional representation*.

A typical example can be the vanilla neural process (NP) (Garnelo et al., 2018b), where the approximate *functional prior* $q_\phi(z|\mathcal{D}^C)$ as *the functional representation* helps induce the predictive function distribution $\mathbb{E}_{q_\phi(z|\mathcal{D}^C)}\left[p(y|z,x)\right]$. Also, the vanilla NP as the functional representation model is the foundation of our developed models and algorithms in this thesis. A large body of our work is to incorporate structural inductive biases into functional representations. This thesis will also present a rethink of the optimization of vanilla NPs and propose a new method to bridge the inference gap.

Our contributions in this thesis are as follows:

- In Chapter (3), a Doubly Stochastic Variational Neural Process (DSVNP) is developed (Wang and Van Hoof, 2020) for multi-output regression and uncertainty quantification in image classification. DSVNP is a type of hierarchical Bayesian model which simultaneously represents functions and captures correlations between inputs and outputs. Experimental results show DSVNP's competitive performance in fitting high dimensional real-world datasets and quantifying the classification uncertainty in a collection of image datasets.

- In Chapter (4), the Mixture of Expert Neural Process (MoE-NP) (Wang and Van Hoof, 2022a) is designed for few-shot regression and meta reinforcement learning problems. MoE-NP is a Bayesian mixture model with both continuous latent variables and categorical latent variables, which handles functions from a mixture of diverse components and tasks in multi-modal distributions. Experimental results show that MoE-NP can distinguish different function components with latent variables and exhibit state-of-art performance in CIFAR10 image completion and continuous control tasks.

- In Chapter (5), the Graph Structured Surrogate Model (GSSM) is proposed together with Amortized Policy Search (APS) (Wang and Van Hoof, 2022b), which applies

to model-based meta reinforcement learning. The method simultaneously learns dynamics models and optimal policies with a few transitions in a model-based meta reinforcement learning setup. Given limited training episodes, the dynamics model with graph structured latent variables can generalize to new MDPs while the learned global latent variable induces task-specific value functions and policies with high performance in Cartpole/Acrobot and Mujoco environments.

- Chapter (6) rethinks the optimization objectives of NPs, analyzes the inference suboptimality, and then solves this within the framework of variational expectation maximization (Wang et al., 2022). The resulting model, referred to as the Self-normalized Importance weighted neural process (SI-NP), can learn the effective functional prior straightforwardly without variational inference and has an improvement guarantee with respect to the log-likelihood of the function dataset. Meanwhile, the conditional neural process (CNP) (Garnelo et al., 2018a) is demonstrated to be equivalent to SI-NPs with a one-sample Monte Carlo estimate. These theoretical claims are verified in the Gaussian process and image completion datasets.

CONTENTS

1 INTRODUCTION 11

1.1 Probabilistic Deep Learning & Meta Learning 11
1.1.1 Uncertainty Quantification 11
1.1.2 Fast Skill Transfer 12
1.2 Notations 12
1.3 Background & Preliminaries 12
1.3.1 Deep Latent Variable Models 12
1.3.2 Functional Representations 14
1.3.3 Variational Inference & ELBO 15
1.4 Research Questions & Technical Contributions 16
1.5 The Layout of this Thesis 19

2 PUBLICATIONS & CONTRIBUTIONS 21

2.1 List of Publications 21
2.2 Software & Repositories 22

3 BAYESIAN HIERARCHICAL FRAMEWORK FOR FUNCTIONAL REPRESENTATION 23

3.1 Introduction 23
3.2 Related Work 24
3.3 Preliminaries 25
3.3.1 Gaussian Processes in the Implicit LVM 26
3.3.2 Neural Processes in the Implicit LVM 26
3.3.3 Connection to Other Models 27
3.4 Methods 28
3.4.1 Neural Process with Hierarchical Latent Variables 28
3.4.2 Approximate Inference and ELBO 29
3.4.3 Scalable Training and Uncertainty-aware Prediction 30
3.4.4 More Insights and Implementation Tricks 31
3.5 Experiments 32
3.5.1 Synthetic Experiments 32
3.5.2 System Identification on Physics Engines 33
3.5.3 Multi-Output Regression on Real-world Dataset 34
3.5.4 Classification with Uncertainty Quantification 35
3.6 Conclusion & Discussion 37

4 MIXTURE OF EXPERTS STRUCTURES FOR FUNCTIONAL REPRESENTATION 39

4.1 Introduction 39
4.2 Related Work 40
4.3 Preliminaries 41

4.3.1 Few-Shot Supervised Learning . . . 41
4.3.2 Meta Reinforcement Learning . . . 42
4.4 Methods . . . 42
4.4.1 Mixture of Expert Neural Processes . . . 43
4.4.2 Scalable Training & Prediction . . . 44
4.4.3 Module Details for Meta Learning . . . 45
4.5 Experiments . . . 49
4.5.1 General Setup . . . 49
4.5.2 Illustration in Toy Regression . . . 50
4.5.3 Few-Shot Supervised Learning . . . 52
4.5.4 Meta Reinforcement Learning . . . 53
4.5.5 Ablation Studies . . . 55
4.6 Conclusion & Discussion . . . 56
5 GRAPH STRUCTURED FUNCTIONAL REPRESENTATION FOR DATA EFFICIENT CONTROL 57
5.1 Introduction . . . 57
5.2 Related Work . . . 58
5.3 Preliminaries . . . 59
5.3.1 Optimization Objective in MBMRL . . . 59
5.3.2 MBMRL with Latent Variables . . . 60
5.3.3 Influence of Model Discrepancy . . . 60
5.4 Methods . . . 61
5.4.1 Graph Structured Latent Variables . . . 61
5.4.2 Approximate Inference & Scalable Training in GSSM . . . 63
5.4.3 Amortized Policy Search . . . 64
5.5 Experiments . . . 67
5.5.1 General Settings . . . 67
5.5.2 Cart-Pole Systems . . . 68
5.5.3 Other Simulation Systems . . . 70
5.6 Conclusion & Discussion . . . 71
6 BRIDGE THE INFERENCE GAPS IN NPS WITH EXPECTATION MAXIMIZATION 73
6.1 Introduction . . . 73
6.2 Related Work . . . 74
6.3 Preliminaries . . . 75
6.4 Optimization Gaps and Statistical Traits . . . 76
6.4.1 Inference Suboptimality in vanilla NPs . . . 76
6.4.2 Evaluation Criteria & Asymptotic Performance . . . 78
6.5 Tractable Optimization via Expectation Maximization . . . 78
6.5.1 Variational Expectation Maximization for NPs . . . 79
6.5.2 Scalable Training and Testing . . . 82
6.6 Experiments . . . 83
6.6.1 Synthetic Regression . . . 83
6.6.2 Image Completion . . . 83

6.7 Conclusion & Discussion 85
7 CONCLUSION 87
8 APPENDICES 89
A Supplementary Materials in Chapter 3 89
A.1 Some Basic Concepts 89
A.2 Proof of Proposition 1 90
A.3 Proof of DSVNP as Exchangeable Stochastic Process 92
A.4 Derivation of Evidence Lower Bound for DSVNP 93
A.5 Implementation Details in Experiments 94
B Supplementary Materials in Chapter 4 99
B.1 Frequently Asked Questions 99
B.2 Probabilistic Graphs in Meta Training/Testing 100
B.3 More Descriptions of NP Family Models and Meta RL 101
B.4 MoE-NPs as Exchangeable $\mathcal{SP}$s 102
B.5 Summary of Existing NP Related Models 103
B.6 Formulation of Evidence Lower Bounds 105
B.7 Experimental Settings and Neural Architectures 110
B.8 Additional Experimental Results 113
C Supplementary Materials in Chapter 5 119
C.1 Frequently Asked Questions 119
C.2 Proof of Lemma 1 121
C.3 Proof of Model Discrepancy 122
C.4 GSSM Modules in PyTorch 123
C.5 Evidence Lower Bound for GSSM 123
C.6 Computational Graphs and Detailed Descriptions 124
C.7 Policy Gradient Estimates in Amortized Policy Search 125
C.8 Experimental Settings and Training Details 127
C.9 Neural Architectures and Parameter Settings 130
D Supplementary Materials in Chapter 6 132
D.1 Probabilistic Generative Process in NPs 132
D.2 Run-time Complexity of Predictive Distributions in GPs & NPs 132
D.3 NPs Formulation & Structural Inductive Biases 132
D.4 Neural Architectures in Implementations 134
D.5 Formulation of Variational Expectation Maximization Method . 134
D.6 Proof of SI-NPs Equivalence with CNPs 137
D.7 Experimental Setup & Implementation Details 139
Bibliography 151
Samenvatting 152

1

INTRODUCTION

1.1 PROBABILISTIC DEEP LEARNING & META LEARNING

In the past decade, with the help of model capacity and computational power increase, deep learning has achieved remarkable progress in computer vision (Krizhevsky et al., 2012), natural language processing (Pennington et al., 2014), sequential decision making (Silver et al., 2017) and so forth. Multiple processing layers in computational models, *e.g.* deep neural networks, enable abstraction of concepts at different levels and significantly improve the state-of-art in related domains due to the compact representation of data (LeCun et al., 2015).

To further release the potential of neural networks, researchers have developed a collection of modules to improve generalization capability, including convolutional networks (LeCun et al., 1995), recurrent networks (Hochreiter and Schmidhuber, 1997), attention networks (Vaswani et al., 2017), residual networks (He et al., 2016), etc. Even so, there remain some limitations that incur doubts from the industry when deploying these models in real-world scenarios. Among them, *uncertainty quantification* and *fast skill transfer* are practical concerns. This thesis primarily concentrates on these two issues. The developed methods are built upon probabilistic deep learning models within a meta learning paradigm.

1.1.1 *Uncertainty Quantification*

Uncertainty quantification is long-standing and challenging. Instead of point estimates, we prefer probabilistic estimates with meaningful confidence intervals in predictions. Since the measured uncertainty reflects the reliability in prediction, it has much use in quantitative finance and the robotics industry. With uncertainty estimates at hand, we can avoid unnecessary risk and make relatively conservative choices in cost-sensitive decision-making (Gal and Ghahramani, 2016).

Such demand inspires the increasing investigation of *probabilistic deep learning* (Ghahramani, 2015). For example, deep Gaussian processes (GPs) (Titsias, 2009) as typical non-parametric models are applied to reason over uncertainties with the help of placed Gaussian priors over functions. Key ingredients in probabilistic deep learning are methods to characterize the different types of uncertainty and utilize uncertainty in decision-making (Gal and Ghahramani, 2016).

1.1.2 *Fast Skill Transfer*

In real-world scenarios, training deep learning models requires heavy computational resources and is time-consuming when the dataset is large-scale. Meanwhile, deep learning models are sensitive to changes in tasks, and the distribution drift inside the dataset can easily cause catastrophic failure using previously trained models. For example, adaptive cruise control is essential in Tesla Autopilot systems (Yaqoob et al., 2019). Cameras in Autopilot capture an enormous amount of images, and the distributions vary from scenario to scenario, *e.g.* different landscapes, buildings, and vehicles in different countries. This reality demands Autopilot's controller to rapidly adapt to new scenarios with a few samples rather than learn policies from scratch.

The above demands drive us to look for learning strategies that can guide models to rapidly adapt to new environments. One plausible way is to make the learning system master the fast skill transfer. Fortunately, *meta learning* or *learning to learn* emerges as a promising paradigm to enable fast deployment of learned models (Finn et al., 2017). This paradigm can also be called *fast learning via slow learning* (Duan et al., 2016) since it seeks a strategy to distill general knowledge from the distribution of tasks to unseen ones. The meta training process is time-consuming because it requires leveraging sufficient past experiences, while the meta testing process of adapting to unseen tasks is relatively fast.

1.2 NOTATIONS

In this section, general math notations are shared across all chapters and are summarized in Table (1). As for other commonly used math notations in separate chapters, I will detail them in corresponding chapters when necessary.

1.3 BACKGROUND & PRELIMINARIES

1.3.1 *Deep Latent Variable Models*

As pointed out in Prof. Max Welling's response to "The Bitter Lesson (Sutton, 2019)" (Welling, 2019), it is challenging to predict without inductive biases. As an indispensable part of learning systems, the role of inductive biases is to restrict the model's hypothesis in a reasonable space, which reflects the understanding of the learning task and impacts the optimization towards a regularized direction.

An inductive bias prevalent in recent years is the existence of unobserved random variables in modeling to explain how data is generated. Deep latent variable models (LVMs) are typical examples. As powerful probabilistic deep learning models, deep LVMs focus on the data generation mechanism. They can achieve prediction with uncertainty quantification.

Here we take variational auto-encoders (VAEs) (Kingma and Welling, 2013; Rezende et al., 2014) as example. Let observations $\mathcal{D} = \{x_1, \ldots, x_n\}$ be a set of independent and identically distributed (*i.i.d.*) random variables and $\forall x_i \in \mathbb{R}^d$, it is assumed that there

Table 1: General Necessary Math Notations in this Thesis. All of these appear in the main thesis and appendices. Some specialized notation will be pointed out in separate chapters.

# Examples	Descriptions
(x_i, y_i)	A single data point with the input x_i and the output y_i
$[x_C, y_C]$	A set of context data points $\{(x_i, y_i)\}$
$[x_T, y_T]$	A set of target data points $\{(x_i, y_i)\}$
$\mathcal{I}_d$	A d-dimensional identity matrix
$\mathbb{R}^k$	A k-dimensional real value vector
diag$[\cdot]$	A diagonal matrix
$\mathcal{O}$	Scale of computational complexity
$\mathcal{D}$	A set of data points
$\mathcal{L}$	An optimization objective function
$\mathcal{M}$	An Markov decision process
$\mathcal{J}$	An expected reward function in RL
$\mathcal{G}$	A structured graph for the model
$< \mathcal{V}, \mathcal{E} >$	Vertices $\mathcal{V}$ and edges $\mathcal{E}$ for a structured graph
τ	A sampled task from a distribution
ρ	An exchangeable stochastic process
$\delta(z - \hat{z})$	A Dirac delta function with a center $\hat{z}$
$k(\cdot, \cdot)$	A kernel function in Gaussian processes
π_φ	A policy function with parameters φ in RL
$p(\cdot)$ or $q(\cdot)$	A probability density function with q to denote the varitional distribution
$\mathcal{N}(\cdot; \mu, \Sigma)$	A Gaussian distribution with a mean vector μ and a covariance matrix Σ
$\texttt{Cat}(\cdot; K, \alpha)$	A categorical distribution with K categories and α vector of probabilities
$D_{KL}[\cdot \parallel \cdot]$	Kullback-Leibler divergence between two probability distributions
$\mathbb{E}[\cdot]$	The expectation function over the random variable
$\mathbb{V}[\cdot]$	The variance function over the random variable
$\mathbb{H}[\cdot]$	The entropy of a random variable's probability distribution
$\sum$	A summation over values of interest
$\lim_{x \to c}[\cdot]$	A limitation operation over variables of interest
∇_ϕ or $\frac{\partial}{\partial \phi}$	Obtation partial derivatives *w.r.t.* the parameter ϕ
$\int [\cdot] dz$	Obtain the integral *w.r.t.* the random variable z
$\circ$	The convolutional operator
$\bigoplus$	Mean poolings over the values of interest

exists a set of k-dimensional unobserved random variables $\mathcal{Z} = \{z_1, \ldots, z_n\}$ as latent variables to explain the observation. Each paired data point (z, x) is generated as follows.

$$\begin{cases} z \sim p(z) \\ p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \Sigma_\theta(z)) \end{cases} \tag{1.1}$$

In this way, the marginal distribution of the observed data point x can be expressed as

$$p_\theta(x) = \int_{\mathbb{R}^k} \underbrace{p(z)}_{\text{prior}} \underbrace{p_\theta(x|z)}_{\text{generative likelihood}} dz = \int_{\mathbb{R}^k} \mathcal{N}(z; 0, \mathcal{I}_k) \mathcal{N}(x; \mu_\theta(z), \Sigma_\theta(z)) dz$$

where a Gaussian distribution $\mathcal{N}(z; 0, \mathcal{I}_k)$ works as the prior in vanilla VAEs. The amortized inference is efficient in learning approximate posteriors of VAEs, which makes it scalable in large datasets. Importantly, the learned latent variable z can be interpreted as the representation of a data point in a lower dimensional space.

1.3.2 *Functional Representations*

In VAEs, as well as some other probabilistic deep learning models, the collected data points are treated as the *i.i.d.* random variables. This hypothesis ignores the existence of correlations between data points, which can be indispensable for reasoning over uncertainty.

In comparison, exchangeable stochastic processes (Ross et al., 1996) in **Definition** (1), *e.g.* Gaussian process latent variable models (GP-LVMs) (Titsias and Lawrence, 2010), attempt to model the joint distribution over all data points and enable the computation of predictive distributions (or equivalently conditional distributions in the output space). This family of models applies to a set of data points. However, it is known that in previous deep stochastic process models, *e.g.* GP-LVMs: (i) the inference method is complex, (ii) the modeled function distribution is restricted to be Gaussian, (iii) the computation of predictive distributions is time-consuming.

Definition 1 (Exchangeable Stochastic Processes) *Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, let $\mu_{x_1,\ldots,x_N}$ be a probability measure on $\mathbb{R}^d$ with $\{x_1, \ldots, x_N\}$ as a finite index set. The defined process is called an exchangeable stochastic process ($\mathcal{SP}$), $\mathcal{S} : X \times \Omega \to \mathbb{R}^d$ such that $\mu_{x_1,\ldots,x_N}(F_1 \times \cdots \times F_N) = \mathbb{P}(\mathcal{S}_{x_1} \in F_1, \ldots, \mathcal{S}_{x_N} \in F_N)$ if it satisfies the exchangeable consistency and marginalization consistency in Kolmogorov extension theorem (Oksendal, 2013).*

Naturally, there arises a question: *How can we extend the inference efficient deep LVM such as VAEs to the function space and learn representations of functions?* The neural process (NP) model well answers this question, and the generative process of NPs are given in **Example** (1). Also, the NP is the foundation of this thesis.

Example 1 (Neural Processes (Garnelo et al., 2018b)) *This family of deep directed latent variable models belongs to exchangeable stochastic processes. A generative process is written as Eq. (1.2) with a global Gaussian latent variable z shared across all data points,*

$$\rho_{x_{1:N}}(y_{1:N}) = \int \underbrace{p(z)}_{\textit{functional prior}} \prod_{i=1}^{N} \underbrace{\mathcal{N}(y_i; \mu_\theta([x_i, z]), \Sigma_\theta([x_i, z]))}_{\textit{generative likelihood}} dz \tag{1.2}$$

where μ_θ and Σ_θ are respectively neural network modules to parameterize mean functions and covariance matrices with the concatenated input $[x_i, z]$.

Concept of Functional Representations. *The summary of context data points $\mathcal{D}^C = \{(x_i, y_i)\}_{i=1}^n$ to represent an underlying function $y = f(x)$* is referred to as the *functional representation*. The context points are partial observations of a function. The *functional representation* is in the form of the *functional prior* and its role is to formulate a distribution of predictive functions.

The functional representation relies on a tractable generative model to characterize the distribution of functions. This significantly distinguishes it from the representation of a single data point in a generative model. In GPs, the learned kernel parameters or distributional parameters specify the functional prior and induce predictive distributions. While in NPs, the latent variable distribution $p(z)$ explains the functional representation straightforwardly. Since the real functional the prior $p(z)$ is intractable, the approximate ones $q_\phi(z|\mathcal{D}^C)$ work as the surrogate to obtain the predictive function distribution $\mathbb{E}_{q_\phi(z|\mathcal{D}^C)}\left[p(y|z,x)\right]$ in NPs.

Connection with Uncertainty Quantification & Fast Skill Transfer. Since the functional representation model, such as the NP, learn distributions over functions, it can provide predictive mean and variances.

When the integral $\mathbb{E}_{q_\phi(z|\mathcal{D}^C)}\left[p(y|z,x)\right]$ is not analytical, uncertainty quantification can be achieved via multiple stochastic forward pass, in the form $z^{(k)} \sim q_\phi(z|\mathcal{D}^C)$ and $y \sim p(y|z^{(k)}, x)$. Finally, we can obtain the Monte Carlo estimate of the mean $\mathbb{E}_{q_\phi(z|\mathcal{D}^C)}\left[p(y|z,x)\right]$ and variance $\mathbb{V}_{q_\phi(z|\mathcal{D}^C)}\left[y|x\right]$ statistics in prediction.

At the same time, the process of obtaining the predictive function distribution with the help of learned functional priors corresponds to the fast skill transfer in the meta testing process.

1.3.3 *Variational Inference & ELBO*

In the presence of deep probabilistic models, inference over distributions of unobserved latent variables is the central object of interest. For example, the computation of the posterior is fundamental in Bayesian inference. Again, the VAE model works as an example to illustrate the relevant concepts. Given the prior distribution $p(z)$ and the generative distribution $p_\theta(x|z)$ in VAEs, the exact form of the posterior is

$$p_\theta(z|x) = \frac{p_\theta(x|z)p(z)}{\int p_\theta(x|z)p(z)dz}.$$

The posterior plays a crucial role in Bayesian prediction or sample generation, *e.g.* $\mathbb{E}_{p_\theta(z|x)}\left[p_\theta(x|z)\right]$. However, we can hardly obtain the exact form when the dimensionality is high in the above denominator integral. Though exact sampling methods with convergence guarantee exist, *e.g.* Markov Chain Monte Carlo, they require expensive sampling and are prohibitive in large-scale high dimensional cases.

As an alternative, approximate methods provide a scalable scheme, and we mainly focus on variational inference (VI) in most of the thesis. Theoretically, we can cast variational inference as an optimization problem, which suits large-scale dataset well.

VI considers a family of parameterized distributions $\mathcal{Q}_\Phi = \{q_\phi(z|x)|\phi \in \Phi\}$ to approximate the real posterior $p_\theta(z|x)$. In optimization, the objective of our interest is reduced to finding $\phi_* = \arg\min_{\phi\in\Phi} D_{KL}\left[q_\phi(z|x) \parallel p_\theta(z|x)\right]$. By expanding such Kullback–Leibler (KL) divergence term, we can obtain the Evidence Lower Bound (ELBO) $\mathcal{L}(\theta,\phi)$ in Eq. (1.3) as a surrogate to maximize.

$$
\begin{aligned}
D_{KL}\left[q_\phi(z|x) \parallel p_\theta(z|x)\right] &= \int q_\phi(z|x) \ln\left[\frac{q_\phi(z|x)}{p_\theta(z|x)}\right] dz \\
&= \int q_\phi(z|x) \ln\left[\frac{q_\phi(z|x)p_\theta(x)}{p_\theta(x|z)p(z)}\right] dz \\
= \int q_\phi(z|x) \ln p_\theta(x) dz - \left[\mathbb{E}_{q_\phi(z|x)}\left[\ln p_\theta(x|z)\right] - D_{KL}\left[q_\phi(z|x) \parallel p(z)\right]\right] & \\
= \ln p_\theta(x) - \mathcal{L}(\theta,\phi) &
\end{aligned}
\tag{1.3}
$$

Since the KL divergence term is non-negative, the log-likelihood $\ln p_\theta(x)$ is lower bounded by the ELBO. Maximization over $\mathcal{L}(\theta,\phi)$ tends to increase the evidence $\ln p_\theta(x)$ when $\mathcal{Q}_\Phi$ is flexible enough. The resulting parameters $\{\phi_*, \theta_*\} = \arg\max \mathcal{L}(\theta,\phi)$ can directly induce the Bayesian prediction function $\mathbb{E}_{q_{\phi_*}(z|x)}\left[p_{\theta_*}(x|z)\right]$.

Though VI is sampling light and has better scalability, it suffers from approximation gaps due to oversimplified parameterized distributions. However, this is not the focus point in this thesis. In **Example** (1), VI is applied to optimize the model (Garnelo et al., 2018b). Also, we utilize VI to obtain tractable optimization objectives for developed models in Chapter (3), Chapter (4) and Chapter (5).

Here we need to tell the difference between VAEs and the functional representation models. Learning functional priors matters more than posterior inference in scenarios to recover the complete function with partial observations (Foong et al., 2020). While in VAEs, prior distributions are mostly fixed as constant and do not work in Bayesian prediction. Chapter (6) will revisit this.

1.4 RESEARCH QUESTIONS & TECHNICAL CONTRIBUTIONS

As mentioned above, learning functional representations is our primary research goal throughout this thesis. Though neural processes provide a simple and elegant way to represent functions, it is non-trivial to develop a universal functional representation model that applies to all scenarios. In this thesis, we refer to constraints on neural network modules or special probabilistic structures for exchangeable stochastic process models as the structural inductive biases.

Apart from the structural inductive biases, the optimization objectives or inference methods for exchangeable stochastic process models are other topics to investigate. This direction has not attracted enough attention, and our preliminary study is a milestone in this domain.

As a result, we tend to split the research questions on exchangeable stochastic processes into ones with available inductive biases and ones concerning general inference problems. The first part corresponds to Chapter (3), Chapter (4) and Chapter (5). We consider incorporating diverse structural inductive biases to learn functional representations and apply variational inference to optimize them. The second part corresponds to Chapter (6). Diverse optimization objectives for functional representations are analyzed, and inference suboptimality inside vanilla neural processes can be resolved with the help of variational expectation maximization. The specific contents in these two parts are around the following research questions.

Note that existing vanilla neural processes for functional representations cannot reasonably handle the learning problems in high-dimensional input/output (I/O) or non-Gaussian output cases. Based on this investigation, we pose the research question as follows.

Research Question 1: *In the traditional pattern recognition problems with high dimensions in I/O, e.g., regression or classification, can we develop more expressive neural process models for prediction with uncertainty quantification?*

To answer this question, we propose DSVNP (Wang and Van Hoof, 2020), a hierarchical variant of neural processes. Our model is built upon the framework of hierarchical Bayes and introduces latent variables at different levels. The global latent variable, similar to its use in NPs, works on the summary of an underlying function, while the introduced local latent variable captures correlations between high dimensional output and the input to improve predictive performance. We combine the inference methods of NPs and conditional VAEs to obtain the tractable objective in optimization. The method can achieve superior performance in real-life multi-output regression and uncertainty quantification in image classification. We refer the reader to Chapter (3) for more details.

Note that previous works assume the studied functions belong to the same family with similar properties and have a simple task distribution, *e.g.* uniform distributions, in the latent space. To overcome the limitations of this assumption, we pose the research question as follows.

Research Question 2: *Considering the complicated data generative processes, e.g., sourced from a mixture of function distributions, can neural processes discover distinguished components and accommodate the functional representations to these cases?*

To answer this question, we propose MoE-NP (Wang and Van Hoof, 2022a), a mixture variant of neural processes. We incorporate the mixture of expert inductive biases into the neural process. As a result, multiple expert latent variables and assignment latent variables participate in generative processes. This design can find a mixture of functional representations as the expert latent variable and categorical variables in identifying appropriate functional components for separate x-domains. Empirically, when the function is discontinuous or multiple co-founders govern the data generation, our

model can achieve state-of-art performance. These are examined in few-shot regression tasks, image completion tasks, and meta reinforcement learning tasks. We refer the reader to Chapter (4) for more details.

Note that the NP can be applied to data efficient control (Galashov et al., 2019). However, using multilayer perceptron (MLP) in neural architectures ignores correlations among transition samples, which weakens NP's performance in predicting physics system dynamics. Also, previous work seldom considers fast adaptation in model-based policy search, which might be critical in time-sensitive control. With these two points, we pose the research question as follows.

Research Question 3: *When the functional presentation meets with model based meta reinforcement learning, can we find functional representations for optimal value functions and policy functions to enable fast deployment in practice?*

To answer this question, we contribute a new meta dynamics model GSSM and amortized policy search (Wang and Van Hoof, 2022b) for model-based meta reinforcement learning missions. Our work is an extension of MLSM (Galashov et al., 2019). Unlike MSLM, our meta learning purpose include (i) learning effective dynamics models in the MDPs family and (ii) learning representations of optimal policies. To achieve this, we develop a variational graph structured dynamics model GSSM with the local latent variable to predict individual transition and the global latent variable to summarize an MDP. Importantly, we amortize the process of seeking model-based optimal policies to induce the optimal policies conditioned on MDP representations. We refer the reader to Chapter (5) for our competitive results and details.

Note that designing various inductive biases or applying neural processes to different domains has attracted increasing attention. However, researchers fail to understand the inference suboptimality inside the vanilla neural process. Significantly, the source of the underfitting phenomenon arising in neural processes and solutions to these questions are less studied. Hence, we pose the research question as follows.

Research Question 4: *Can we build up connections between optimization objectives of functional representation models, analyze the convergence results, and seek new optimization methods with improvement guarantees for these models?*

To answer this question, we rethink the optimization objectives of neural processes. After analyzing the relation between approximate objectives in neural processes and evidence lower bounds, we find the optimization in vanilla NPs cannot guarantee a convergence towards local optimality in the likelihood of meta datasets. Hopefully, this can be resolved by designing a new surrogate function for neural processes. With the help of an importance sampling trick, we can obtain SI-NP with a likelihood improvement guarantee (Wang et al., 2022). Meanwhile, we perform the limit analysis to demonstrate that the conditional neural process is an example of SI-NP with one Monte Carlo sample. Our learned functional prior seldom collapse to a Dirac delta function and can retain satisfying uncertainty quantification performance in typical benchmarks. We refer the reader to Chapter (6) for more details.

1.5 THE LAYOUT OF THIS THESIS

To make connections between chapters clear, we have the diagram next page. Following are paragraphs concerning chapters of models/algorithms.

Chapter (3), Chapter (4), and Chapter (5) are parallel in the organization. We mainly contribute new functional representation models in this part. The developed DSVNPs, MoE-NPs, GSSM-APS correspond to different structural inductive biases for functional representations and hence apply to different scenarios. Variational inference is utilized to optimize these models.

Chapter (6) independently focuses on optimization objectives and inference methods. We contribute an expectation maximization based inference algorithm for functional representation models and report the resulting functional representation model SI-NPs. This inference algorithm can be combined with all structural inductive biases, not limited to those in Chapter (3), Chapter (4), and Chapter (5).

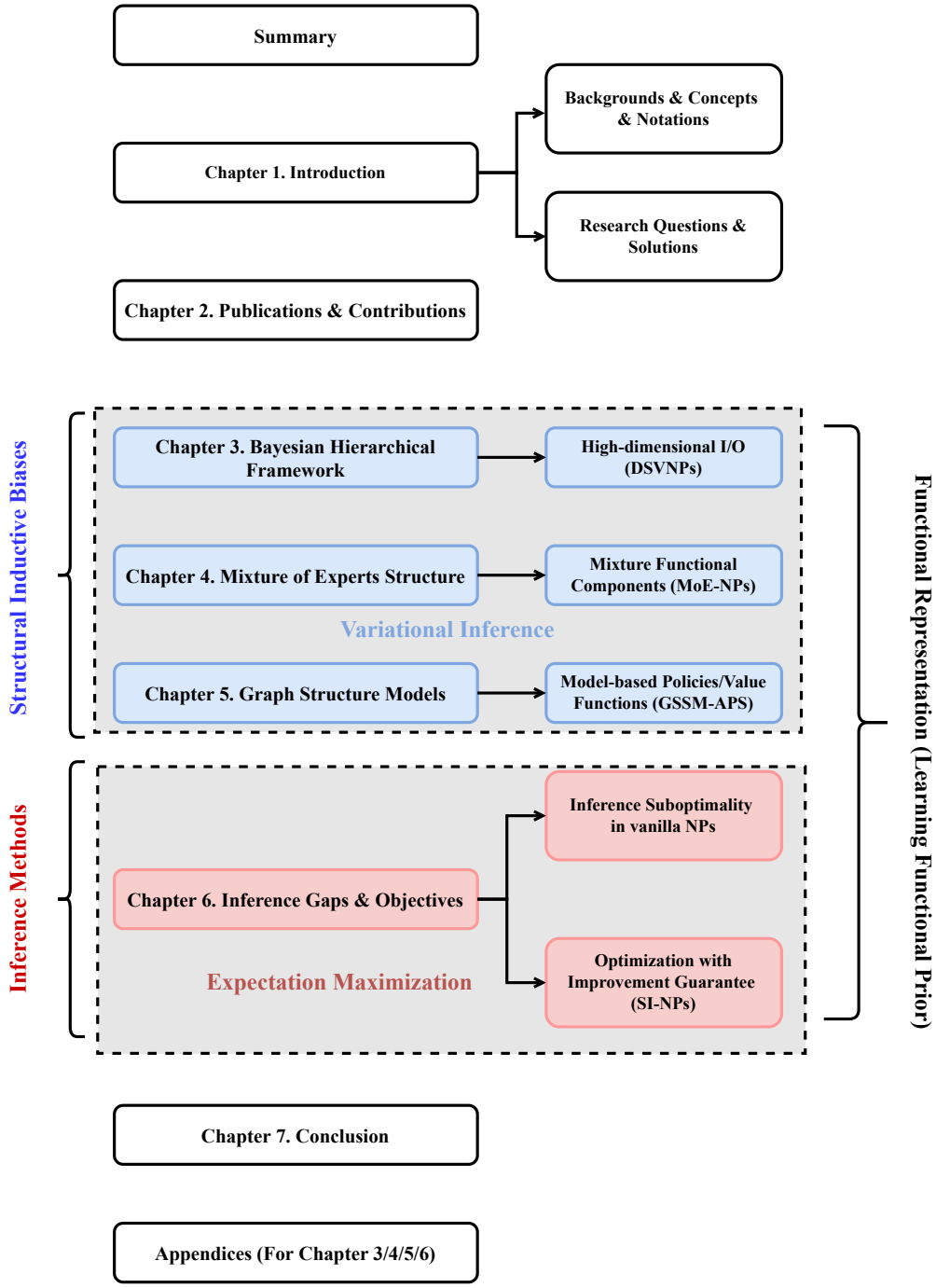


Figure 1: Overview of the Thesis Layout.

2

PUBLICATIONS & CONTRIBUTIONS

2.1 LIST OF PUBLICATIONS

The following publications and work in progress constitute this Ph.D. thesis. These research outputs were done when Qi Wang pursued a Ph.D. at AMLab from June 2019 to October 2022. Note that Chapters (3)/(4)/(5)/(6) are respectively built upon the publications (Wang and Van Hoof, 2020)/(Wang and Van Hoof, 2022a)/(Wang and Van Hoof, 2022b)/(Wang et al., 2022).

- Qi Wang, Herke van Hoof. Doubly Stochastic Variational Inference for Neural Processes with Hierarchical Latent Variables. *International Conference on Machine Learning, PMLR, 2020*. `http://proceedings.mlr.press/v119/wang20s/wang20s.pdf`.

- Qi Wang, Herke van Hoof. Learning Expressive Meta-Representations with Mixture of Expert Neural Processes. *International Conference on Neural Information Processing Systems, 2022*. Final Version: `https://drive.google.com/file/d/1LfR43Cr8TliZcwqErhT7nsAOK296BZ0g/view?usp=sharing`.

- Qi Wang, Herke van Hoof. Model-based Meta Reinforcement Learning using Graph Structured Surrogate Models and Amortized Policy Search. *International Conference on Machine Learning, PMLR, 2022*. `https://proceedings.mlr.press/v162/wang22z/wang22z.pdf`.

- Qi Wang, Marco Federici, Herke van Hoof. Bridge the Inference Gaps of Neural Processes via Expectation Maximization. In preparation for *International Conference on Learning Representations 2023*.

Meanwhile, we specify the contributions of participants in this project. Qi Wang contributed original ideas, algorithm design, experimental examination, and paper writing in all the above publications. Herke van Hoof provided important insights, feedback and suggestions on the written manuscripts. For the work "Bridge the Inference Gaps of Neural Processes via Expectation Maximization", Marco Federici provided helpful discussions and feedback on the submission draft. He also contributes an explanation of SI-NPs from an information theoretical perspective in the future version, which is however not covered in this thesis. The whole research project received consistent support from Max Welling.

2.2 SOFTWARE & REPOSITORIES

Overall, the first author in the above publications contributes python packages and GitHub repositories for the corresponding models & algorithms developed in publications. Details are as follows.

- The implementation of DSVNP can be easily found in the Appendix file.
- Two examples of MoE-NPs implementations are listed as follows
  `https://github.com/codeanonymous233/ICMoENP` for image completion
  and `https://github.com/codeanonymous233/MoENP` for meta RL.
- The implementation of GSSM+APS can be found in
  `https://github.com/codeanonymous233/anonymouscode`.
- The implementation of SI-NPs will be uploaded to
  `https://github.com/codeanonymous233/SINP` after the final acceptance.

Note that all codes will be further polished and updated to Qi Wang's personal Github repository `https://github.com/hhq123gogogo`.

3

BAYESIAN HIERARCHICAL FRAMEWORK FOR FUNCTIONAL REPRESENTATION

In this chapter, we investigate the use of the Bayesian hierarchical framework in neural processes and introduce a new variant of a neural process model. With the help of hierarchical latent variables, our model can address functional representation problems with the high dimensional input/output and achieve satisfying performance.

3.1 INTRODUCTION

Placing distributions over functions has a lot of advantages. It can characterize the underlying uncertainties from observations and enable reliable decision-making. For example, the uncertainty-aware dynamics model enjoys popularity in model-based reinforcement learning, and Gaussian processes (GPs) deployed in PILCO enable the propagation of uncertainty in forecasting future states (Deisenroth and Rasmussen, 2011). Another specific instance can be found in demonstration learning; higher uncertainty in prediction would suggest the learning system to query new observations to avoid dangerous behaviors (Thakur et al., 2019).

In the past few years, a variety of models combining GPs and deep neural networks have been proposed (Salimbeni and Deisenroth, 2017; Snelson and Ghahramani, 2006; Titsias, 2009; Titsias and Lawrence, 2010). However, there still remain some concerns in GP induced predictive distributions. One is high computational complexity in prediction due to the matrix inversion, and another is less flexibility in function space. Recognized as an explicit stochastic process model, the vanilla GP strongly depends on the assumption that the joint distribution is Gaussian. The computational and scalable issues facilitate the birth of adaptations or approximate variants for GP related models (Garnelo et al., 2018a,b; Louizos et al., 2019), which incorporate latent variables in modeling to account for uncertainties.

Research Motivations. As an alternative for GPs, the neural process (NP) can characterize function distributions. By learning a global latent variable, NPs are able to obtain the predictive distribution at the cost of lower computations. However, such a model is known to suffer from underfitting (Kim et al., 2019a) and cannot sufficiently handle cases when variables' input/output are high dimensional.

Developed Methods. We investigate NP related models with a unified model and explore more expressive approximations toward general stochastic processes ($\mathcal{SP}$s). To this end, hierarchical latent variables, respectively global and target specific local latent variables,

are considered part of the model structure, which helps improve flexibility in predictive distributions. In this way, the novel variational approximate model for NPs has the potential to solve learning problems in high-dimensional cases.

Our primary contributions are two-fold in this chapter:

- We systematically revisit NPs, $\mathcal{SP}$s, and other related models from a unified perspective with an implicit Latent Variable Model (LVM). Both GPs and NPs are studied in this hierarchical LVM.
- The Doubly Stochastic Variational Neural Process (DSVNP) is proposed to enrich the NP family. Experimental results demonstrate the effectiveness of the proposed Bayesian model in high dimensional domains, including regression with multiple outputs and uncertainty-aware image classification.

3.2 RELATED WORK

Scalability and Expressiveness in Stochastic Process. GPs are the most well-known member of $\mathcal{SP}$s family and have inspired many extensions, such as deep kernel learning (Wilson et al., 2016a,b) and sparse GPs (Snelson and Ghahramani, 2006) with better scalability. Especially, the latter incorporated sparse prior in function distribution and utilized a small proportion of observations in predictions. In multi-task cases, several GP variants were proposed (Moreno-Muñoz et al., 2018; Bonilla et al., 2008; Zhao and Sun, 2016). Other works also achieve sparse effect but with variational inference, approximating the posterior in GPs and optimizing ELBO (Hensman et al., 2015; Salimbeni et al., 2019; Titsias and Lawrence, 2010). Another branch is about directly capturing uncertainties with deep neural networks, which is revealed in NP related models. Other extensions include generative query network (Eslami et al., 2018), sequential NP (Singh et al., 2019) and convolutional conditional NP (Gordon et al., 2020b). Variational implicit process (Ma et al., 2019) targeted at more general $\mathcal{SP}$s and utilized GPs in latent space as approximation. Sun et al. (2019) proposed functional variational Bayesian neural networks, and variational distribution over functions of measurement set was used to represent $\mathcal{SP}$s. The more recently proposed functional NPs (Louizos et al., 2019) characterized a novel family of exchangeable stochastic processes, placing more flexible distributions over latent variables and constructing directed acyclic graphs with latent affinities of instances in inference and prediction.

Uncertainty Quantification and Computational Complexity. GPs can well characterize aleatoric uncertainty and epistemic uncertainty through kernel function and Gaussian noise. Nevertheless, $\mathcal{SP}$s with non-Gaussian marginals are crucial in modeling. Apart from GPs, some other techniques exist, such as Dropout (Gal and Ghahramani, 2016) or other variants of Bayesian neural networks (Louizos et al., 2017) to quantify uncertainty. In (Depeweg et al., 2018), uncertainties were further decomposed in a Bayesian neural network. DSVNP can theoretically capture both uncertainties as an approximate prediction model for general $\mathcal{SP}$s and approaches the problem in a Bayesian way. For the computational cost in prediction, the superior sparse GPs with K-rank covariance matrix approximations (Burt et al., 2019) are with the complexity $\mathcal{O}((M+N)K^2)$,

while the variants of CNPs or NPs mostly reduce the complexity $\mathcal{O}((N+M)^3)$ in GPs to $\mathcal{O}(M+N)$ in the prediction process. And those for AttnNP and DSVNP are $\mathcal{O}((M+N)N)$.

3.3 PRELIMINARIES

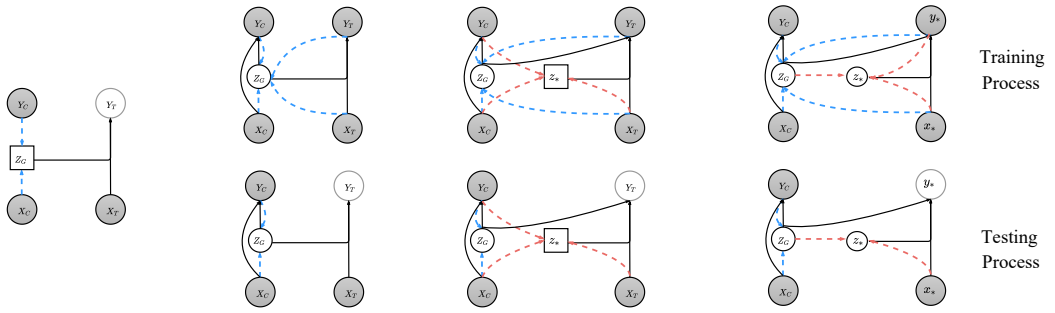


Figure 2: Probabilistic Graphs for CNP, vanilla NP, Attentive NP and DSVNP. The blue dotted lines characterize the inference towards global latent variable z_G, while the pink dotted lines are for target specific local latent variables z_*. The ones in the first row are training cases, while those in the second row are testing cases.

Generally, a stochastic process places a distribution over functions, and any finite collections of variables can be associated with an implicit probability distribution. Here, we naturally formulate an implicit LVM (Rezende et al., 2014; Kingma and Welling, 2014) to characterize General Stochastic Function Processes (GSFPs). The conceptual generation paradigm for this LVM can be depicted in the following equations,

$$z_i = \phi(x_i) + \epsilon(x_i) \tag{3.1a}$$
$$y_i = \varphi(x_i, z_i) + \zeta_i \tag{3.1b}$$

where terms ϵ and ζ respectively indicate the stochastic component in the latent space and random noise in observations. To avoid ambiguity in notation, we declare the stochastic term ϵ as an index dependent random variable $\epsilon(x_i)$, and ζ_i is observation noise in the environment. Also, the transformations ϕ and φ are assumed to be Borel measurable, and the latent variables in Eq. (3.1a) are not restricted. Note that they can be some set of random variables with statistical correlations without loss of generality.

When Kolmogorov Extension Theorem (Oksendal, 2013) is satisfied for $\epsilon(x_i)$, a latent $\mathcal{SP}$ can be induced. Eq. (3.1a) decomposes the process into a deterministic and stochastic component. The transformation φ in Eq. (3.1b) is directly connected to the output. Such a generative process can simultaneously inject *aleatoric uncertainty* and *epistemic uncertainty* in modelling (Hofer et al., 2002), but inherent correlations in examples make the exact inference intractable mostly.

Another principal issue is a prediction with permutation invariance, which learns a conditional distribution in $\mathcal{SP}$ models. With the context $C = \{(x_i, y_i) | i = 1, 2, \ldots, N\}$

and input variables of the target x_T, we seek a stochastic function f_θ mapping from X to Y and formalize the distribution as $p_\theta(y_T|x_C, y_C, x_T)$[1] invariant to the order of context observations. The definitions of permutation invariant functions (PIFs) and permutation equivariant functions (PEFs) are included in Appendix (A.1).

3.3.1 *Gaussian Processes in the Implicit LVM*

Let us consider a trivial case in the LVM when the operation φ is an identity map, ζ is Gaussian white noise, and the latent layer follows a multivariate Gaussian distribution. This degenerated case indicates a GP, and terms ϕ, ϵ are respectively the mean function and the zero-mean GP prior. Meanwhile, recall that the prediction at target input x_T in GPs relies on a predictive distribution $p(y_T|x_C, y_C, x_T)$, where the mean and covariance matrix are inferred from the context $[x_C, y_C]$ and target input x_T.

$$\begin{aligned} \mu(x_T; x_C, y_C) &= \phi_\theta(x_T) + \Sigma_{T,C}\Sigma_{C,C}^{-1}\big(y_C - \phi_\theta(x_C)\big) \\ \Sigma(x_T; x_C, y_C) &= \Sigma_{T,T} - \Sigma_{T,C}\Sigma_{C,C}^{-1}\Sigma_{C,T} \end{aligned} \tag{3.2}$$

Here ϕ_θ and Σ in Eq. (3.2) are vectors of mean functions and covariance matrices. For additive terms, they embed context statistics and connect them to the target sample x_T. Furthermore, two propositions are drawn, which we prove in Appendix (A.2).

Proposition 1 *The statistics of GP predictive distributions, such as mean and (co)-variances, for a specific point x_* are permutation invariant functions, while those in $p(y_T|x_C, y_C, x_T)$ are permutation equivariant functions.*

3.3.2 *Neural Processes in the Implicit LVM*

In non-GP scenarios, inference and prediction processes for the LVM can be non-trivial, and NPs are the family of approximate models for implicit $\mathcal{SP}$s. Also, relationship between GPs and NPs can be explicitly established with deep kernel network (Rudner et al., 2018). Note that *NPs translate some properties of GPs to predictive distributions, especially permutation invariance of context statistics*, which is highlighted in **Proposition** (1). Here three typical models are investigated, respectively conditional neural process (CNP) (Garnelo et al., 2018a),vanilla NP (Garnelo et al., 2018b) and attentive neural process (AttnNP) (Kim et al., 2019b).

When approximate inference is applied to NP family with Latent Variables, a preliminary evidence lower bound (ELBO) for the training process can be derived, which aims at predictive distributions for most NP related models.

$$\begin{aligned} \ln\Big[p(y_T|x_C, y_C, x_T)\Big] \geq\ & \mathbb{E}_{q_\phi(z_T|x_C,y_C,x_T,y_T)} \ln\Big[p_\theta(y_T|x_T, z_T)\Big] \\ & -D_{KL}\Big[q_\phi(z_T|x_C, y_C, x_T, y_T) \parallel p(z_T|x_C, y_C, x_T)\Big] \end{aligned} \tag{3.3}$$

1 For brief notations, the inputs, outputs of the context and the target are respectively denoted as $x_C = x_{1:N}$, $y_C = y_{1:N}$, $x_T = x_{1:N+M}$, $y_T = y_{1:N+M}$. Only in CNP, $x_T = x_{N+1:N+M}$, $y_T = y_{N+1:N+M}$. And $[x_*, y_*]$ refers to any instance in the target.

To ensure context information invariant to orders of points, CNP embeds the context point $[x_C, y_C]$ in an elementwise way and then aggregates them with a permutation invariant operation $\oplus$, such as mean or max pooling.

$$r_i = h_\theta(x_i, y_i), \quad r_C = \bigoplus_{i=1}^{N} r_i \tag{3.4}$$

The latent variable in CNP is a deterministic embedding of the form $p_\theta(z_C|x_C, y_C) = r_C(x_C, y_C)$. Following Eq. (3.1b), CNP decodes statistics as the mean and the variance for a predictive distribution.

For vanilla NPs, the encoder structure resembles CNPs, and the learned embedding variable r in Eq. (3.4) is no longer a function but a Gaussian variable after amortized transformations. For the graphical structure of this LVM in vanilla NPs (Refer to Figure 2 (b)), all latent variables are degraded to a global Gaussian latent variable, which accounts for the consistency.

AttnNP further improves the expressiveness of context information in NP, leaving the latent variable as the combination of a global variable and a local variable. Especially, the attention network uses self-attention or dot-product attention to enable transformations of context points and the extraction of hierarchical dependencies between context points and target points. For the graphical model of this LVM in AttnNP, the context information is instance-specific. The latent variable of AttnNP in Eq. (3.5) is the concatenation of attention embedding z_{attn} from element-wise context embedding s_i and a global latent variable z_G drawn from an amortized distribution parameterized with Eq. (3.4).

$$z_{attn} = \bigoplus_{i=1}^{N} w(x_i, x_*)s_i, \quad z = [z_{attn}, z_G] \tag{3.5}$$

In summary, AttnNP boosts performance with attention networks, which implicitly seek more flexible functional translations for each target.

3.3.3 *Connection to Other Models*

In some scenarios, when the latent layer in Eq. (3.1a) is specified as a Markovian chain, the LVM degrades to the classical state space model. If random variables in the latent layer of the LVM are independent, the resulted neural network is similar to the conditional variational auto-encoder (Sohn et al., 2015), and no context information is utilized for prediction. Instead, correlations between latent variables in the hidden layer increase the model capacity.

The induced $\mathcal{SP}$ in Eq. (3.1b) is a warped GP when the latent $\mathcal{SP}$ is a GP and the transformation φ is nonlinear monotonic (Snelson et al., 2004). In addition, several previous works integrate this idea in modeling as well, and representative examples are deep GPs (Dai et al., 2016a) and hierarchical GPs (Tran et al., 2016).

3.4 METHODS

3.4.1 *Neural Process with Hierarchical Latent Variables*

In the last section, we gain more insights into the mechanism of GPs and NPs and disentangle these models with the implicit LVM. A conclusion can be drawn that the posterior inference conditioned on the context requires both approximate distributions with permutation invariance and some bridge to connect observations and the target in latent space. Note that the global induced latent variable may be insufficient to describe dependencies, and critical challenge comes from *non-stationarity* and *locality*, which are even crucial in high-dimensional cases.

Table 2: Structure Summary over NP Related Models on Training Dataset. Here f corresponds to some functional operation. Global latent variable in CNP only governs points to predict, while that in NP works for the whole points. Furthermore, local latent variables in AttnNP and DSVNP are distinguished, with the latter as a latent random variable.

NP Family	Recognition Model	Generative Model	Prior Distribution	Latent Variable
CNP	$z_C = f(x_C, y_C)$	$p(y_T \mid z_C, x_T)$	NULL	Global
NP	$q(z_G \mid x_C, y_C, x_T, y_T)$	$p(y_T \mid z_G, x_T)$	$p(z_G \mid x_C, y_C)$	Global
AttnNP	$q(z_G \mid x_C, y_C, x_T, y_T)$, $z_* = f(x_C, y_C, x_*)$	$p(y_* \mid z_G, z_*, x_*)$	$p(z_G \mid x_C, y_C)$	Global +Local
DSVNP (Ours)	$q(z_G \mid x_C, y_C, x_T, y_T)$, $q(z_* \mid z_G, x_*, y_*)$	$p(y_* \mid z_G, z_*, x_*)$	$p(z_G \mid x_C, y_C)$, $p(z_* \mid z_G, x_*)$	Global +Local

Hence, we present a hierarchical way to modify NPs, and the trick is to involve auxiliary latent variables for NPs and derive a new evidence lower bound for different levels of random variables with doubly stochastic variational inference (Salimbeni and Deisenroth, 2017; Titsias and Lázaro-Gredilla, 2014). The original intention of involving auxiliary latent variables is to improve the expressiveness of approximate posteriors, and it is commonly used in deep generative models (Maaløe et al., 2016). So, as displayed in Table (2), DSVNP considers a global latent variable and a local latent variable to convey context information at different levels. Our work is also consistent with the hierarchical implicit Bayesian neural networks (Tran et al., 2016, 2017), which distinguish the role of latent variables and induce more powerful posteriors. Without exception, the local latent variable z_* refers to any data point (x_*, y_*) for prediction in DSVNP in the remainders of this paper.

To extract hierarchical context information for the predictive distribution, we distinguish the global latent variable and the local latent variable in a Bayesian model, and the induced LVM is DSVNP. This variant shares the same prediction structure with AttnNP. The global latent variable is shared across all observations, and the role of context points resembles inducing points in sparse GP (Snelson and Ghahramani, 2006), summarizing general statistics in the task. As for the local latent variable in our proposed DSVNP, it is an auxiliary latent variable responsible mainly for variations of instance locality. From another perspective, DSVNP combines the global latent variable in vanilla

NPs with the local latent variable in conditional variational autoencoder (C-VAE). This implementation in model construction separates the global and sample-specific variations and theoretically increases the expressiveness of the neural network.

As illustrated in Fig. (2.d), the target to predict is governed by these two latent variables. The global latent variable z_G and the local latent variable z_* contribute to the prediction. Formally, the generative model as a $\mathcal{SP}$ is described as follows, where exact inferences for latent variables z_G and z_* are infeasible.

$$\rho_{x_{1:N+M}}(y_{1:N+M}) = \iint p(z_G) \prod_{i=1}^{N+M} p(y_i|z_G, z_i, x_i) p(z_i|x_i, z_G) dz_{1:N+M} dz_G \tag{3.6}$$

Meanwhile, we emphasize that this generation method naturally induces an *exchangeable stochastic process* (Bhattacharya and Waymire, 2009). The proof is given in Appendix (A.3).

3.4.2 *Approximate Inference and ELBO*

With the relationship between these variables clarified, we can characterize the inference process for DSVNP, and then a new ELBO is presented. Distinguished from AttnNP, we need to infer global and local latent variables with evidence from the collected dataset. Posteriors of the global and local latent variables on the training dataset are approximated with distributions like vanilla NPs, mapping Eq. (3.4) to means and variances. And inference towards local latent variables requires target information in the approximate posterior,

$$q_{\phi_{1,1}} = \mathcal{N}\Big(z_G|\mu(x_C, y_C, x_T, y_T), \Sigma(x_C, y_C, x_T, y_T)\Big) \tag{3.7}$$

$$q_{\phi_{2,1}} = \mathcal{N}\Big(z_*|\mu(z_G, x_*, y_*), \Sigma(z_G, x_*, y_*)\Big) \tag{3.8}$$

where $q_{\phi_{1,1}}$ and $q_{\phi_{2,1}}$ are approximate posteriors in training process. The generative process is reflected in Eq. (3.9) for DSVNP, where g_θ indicates a decoder in a neural network.

$$p(y_*|x_C, y_C, x_*) = g_\theta(z_G, z_*, x_*) \tag{3.9}$$

Consequently, this difference between vanilla NP and DSVNP leads to another ELBO or negative variational free energy $\mathcal{L}$ as the right term,

$$\begin{aligned} \ln\Big[p(y_*|x_C, y_C, x_*)\Big] \geq \mathbb{E}_{q_{\phi_{1,1}}} \mathbb{E}_{q_{\phi_{2,1}}} \ln[p(y_*|z_G, z_*, x_*)] \\ -\mathbb{E}_{q_{\phi_{1,1}}}[D_{KL}[q_{\phi_{2,1}}(z_*|z_G, x_*, y_*) \parallel p_{\phi_{2,2}}(z_*|z_G, x_*)]\Big] \\ -D_{KL}\Big[q_{\phi_{1,1}}(z_G|x_C, y_C, x_T, y_T) \parallel p_{\phi_{1,2}}(z_G|x_C, y_C)\Big] \end{aligned} \tag{3.10}$$

where $p_{\phi_{1,2}}(z_G|x_C, y_C)$ and $p_{\phi_{2,2}}(z_*|z_G, x_*)$ parameterized with neural networks are used as prior distributions. Here we no longer employ standard normal distributions with

Algorithm 1: Variational Inference for DSVNP in Training.

Input: Dataset $\mathcal{D}$, Maximum context points N_{max}, Learning rate α, Batch size B and Epoch number m.
Output: Model parameters ϕ_1, ϕ_2 and θ.
Initialize parameters ϕ_1, ϕ_2, θ of recognition model and generative model in Eq. (3.7), (3.8) and (3.9).
for $i = 1$ **to** m **do**
  Draw some context number $N_C \sim U[1, N_{max}]$;
  Draw mini-batch instances and formulate context-target pairs $\{(x_C, y_C, x_T, y_T)_{bs}\}_{bs=1}^{B} \sim \mathcal{D}$;
  Feedforward the mini-batch instances to recognition model q_{ϕ_1}:
    Draw sample of latent variable $z_G \sim q_{\phi_{1,1}}$ in Eq. (3.7);
    Draw sample of latent variable $z_* \sim q_{\phi_{2,1}}$ in Eq. (3.8);
  Feedforward the latent variables to discrimination model p_θ:
    Compute conditional probability distribution in Eq. (3.9);
  Update parameters by Optimizing Eq. (3.11):
    $\phi_1 \leftarrow \phi_1 - \alpha \nabla_{\phi_1} \mathcal{L}_{MC}$ with $\phi_1 = [\phi_{1,1}, \phi_{1,2}]$;
    $\phi_2 \leftarrow \phi_2 - \alpha \nabla_{\phi_2} \mathcal{L}_{MC}$ with $\phi_2 = [\phi_{2,1}, \phi_{2,2}]$;
    $\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{MC}$;
end for

zero prior information. Instead, these are parameterized with two diagonal Gaussians for simplicity and learned in an amortized way.

3.4.3 *Scalable Training and Uncertainty-aware Prediction*

Based on the inference process in DSVNP and the corresponding ELBO in Eq. (3.10), the Monte Carlo estimation for the negative lower bound is derived, in which we wish to minimize,

$$\mathcal{L}_{MC} = -\frac{1}{K}\sum_{k=1}^{K}\Big[\frac{1}{S}\sum_{s=1}^{S}\ln[p(y_*|x_*, z_*^{(s)}, z_G^{(k)})] - D_{KL}[q(z_*|z_G^{(k)}, x_*, y_*) \parallel p(z_*|z_G^{(k)}, x_*)]\Big] + D_{KL}\Big[q(z_G|x_C, y_C, x_T, y_T) \parallel p(z_G|x_C, y_C)\Big] \tag{3.11}$$

where latent variables are sampled as $z_G^{(k)} \sim q_{\phi_{1,1}}(z_G|x_C, y_C)$ and $z_*^{(s)} \sim q_{\phi_{2,1}}(z_*|z_G^{(k)}, x_*, y_*)$. Moreover, the resulting Eq. (3.11) is employed as the objective function in the training process. To reduce variance in sampling, Kingma and Welling (2014) introduces the reparameterization trick for all approximate distributions, and the model is optimized using Stochastic Gradient Variational Bayes (Kingma and Welling, 2014). More details can be found in the Algorithm (1).

The predictive distribution is of our interest. For DSVNP, prior networks as $p(z_G|x_C, y_C)$ and $p(z_*|z_G, x_*)$ are involved in prediction, and this leads to the integration over both global and local latent variables here as revealed in Eq. (3.12).

$$p(y_*|x_C, y_C, x_*) = \iint p(y_*|z_G, z_*, x_*) p_{\phi_{1,2}}(z_G|x_C, y_C) p_{\phi_{2,2}}(z_*|z_G, x_*) dz_G dz_* \quad (3.12)$$

For uncertainty-aware prediction, there exist different approaches for Bayesian neural networks. Generally, the conditional distribution in neural networks can be derived once the model is well trained. The accuracy can be evaluated through deterministic inference over latent variables, i.e., $\tilde{z_G} = \mathbb{E}[z_G|x_C, y_C]$, $\tilde{z_*} = \mathbb{E}[z_*|z_G, x_*]$, $y_* = \arg\max_y p(y|\tilde{z_*}, z_G\tilde{}, x_*)$.

The Monte Carlo estimation over Eq. (3.12), which is commonly used for prediction, can be written in the following equation,

$$p(y_*|x_C, y_C, x_*) \approx \frac{1}{KS} \sum_{k=1}^{K} \sum_{s=1}^{S} p_\theta(y_*|x_*, z_*^{(s)}, z_G^{(k)}) \quad (3.13)$$

where the global and local latent variables are sampled in prior networks through ancestral sampling as $z_G^{(k)} \sim p_{\phi_{1,2}}(z_G|x_C, y_C)$ and $z_*^{(s)} \sim p_{\phi_{2,2}}(z_*|z_G^{(k)}, x_*)$.

3.4.4 *More Insights and Implementation Tricks*

The global and local latent variables govern different prediction and sample generation variations. This is a part of the motivations for AttnNP and DSVNP. Interestingly, the inference for our induced $\mathcal{SP}$ integrates the aspects of vanilla NPs (Eslami et al., 2018) and C-VAEs (Sohn et al., 2015).

Similar to β-VAE (Higgins et al., 2017), we rewrite the right term in Eq. (3.10) with constraints and these restrict the search for variational distributions. Equivalently, tuning the weights of divergence terms in Eq. (3.10) leads to a varying balance between global and local information.

$$\begin{aligned} &\max_{\phi_1, \phi_2, \theta} \mathbb{E}_{q_{\phi_{1,1}}} \mathbb{E}_{q_{\phi_{2,1}}} \ln[p_\theta(y_*|z_G, z_*, x_*)] \\ &D_{KL}\Big[q(z_G|x_C, y_C, x_T, y_T) \parallel p(z_G|x_C, y_C)\Big] < \epsilon_G \\ &\mathbb{E}_{q_{\phi_{1,1}}}\Big[D_{KL}[q(z_*|z_G, x_*, y_*) \parallel p(z_*|z_G, x_*)]\Big] < \epsilon_L \end{aligned} \quad (3.14)$$

Here, a more practical objective in implementations derived from weight calibrations in Eq. (3.15).

$$\begin{aligned} \mathcal{L}_{MC}^{W} = -\frac{1}{K} \sum_{k=1}^{K} \Big[\frac{1}{S} \sum_{s=1}^{S} \ln[p(y_*|x_*, z_*^{(s)}, z_G^{(k)})] - \beta_1 D_{KL}[q(z_*|z_G^{(k)}, x_*, y_*) \parallel p(z_*|z_G^{(k)}, x_*)] \Big] \\ -\beta_2 D_{KL}\Big[q(z_G|x_C, y_C, x_T, y_T) \parallel p(z_G|x_C, y_C)\Big] \end{aligned} \quad (3.15)$$

Also, training stochastic model with multiple latent variables is non-trivial, and several works about KL divergence term annealing (Sønderby et al., 2016) or dynamically

adapting for the weights exist. Notably, the target-specific KL divergence term is sometimes suggested to assign more penalty to guarantee the consistency between approximate posterior and prior distribution (Kohl et al., 2018; Sohn et al., 2015).

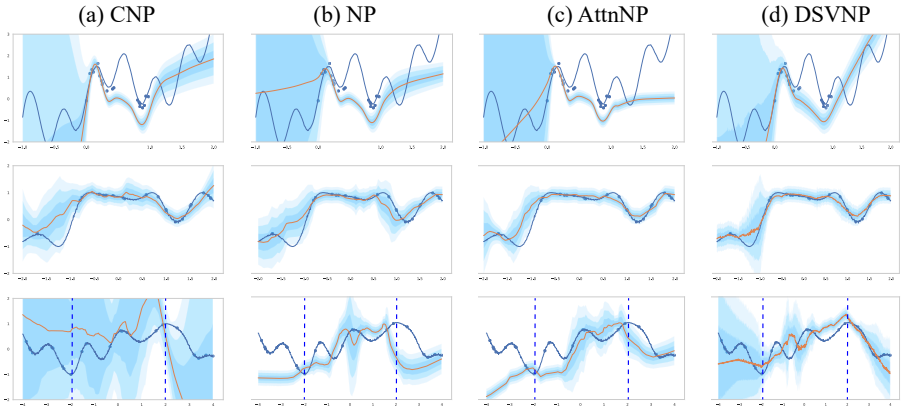


Figure 3: Function Prediction in Interpolation and Extrapolation. Blue curves are ground truth with dotted units as context points, and orange ones are predicted results. Rows from up to down indicate cases: single function with noise, interpolation, and extrapolation towards realizations from a stochastic process. The shadow regions are ±3 standard deviations from the mean.

3.5 EXPERIMENTS

This section starts with learning predictive functions on several toy datasets. Then high-dimensional tasks, including system identification on physics engines, multioutput regression on the real-world dataset, and image classification with uncertainty quantification, are performed to evaluate the properties of NP related models. The dot-product attention is used in all AttnNPs. All implementation details are attached in Appendix (A.5).

3.5.1 *Synthetic Experiments*

We initially investigate the *episdemic* uncertainty captured by NP related models on a 1-d regression task, and the function (Osband et al., 2016a) is characterized as $y = x + \epsilon + \sin(4(x+\epsilon)) + \sin(13(x+\epsilon))$. Observations as the training set include 12 points and 8 points respectively uniformly drawn from intervals $U[0, 0.6]$ and $U[0.8, 1.0]$, with the noise drawn from $\epsilon \sim N(0, 0.003^2)$.

As illustrated in the first row of Fig. (3), we can observe that CNP and DSVNP better quantify variance outside the interval $[0, 1.0]$. At the same time, AttnNP either overestimates or underestimates the uncertainty to show higher or lower standard deviations in regions with fewer observations. All models share similar properties with GPs in predictive distributions, displaying lower variances around observed points. As for the gap in interval $[0.6, 0.8]$, the revealed uncertainty is consistent to that in (Sun et al., 2019; Hernández-Lobato and Adams, 2015) with intermediate variances.

Table 3: Average Negative Log-likelihoods over all target points on realizations from Synthetic Stochastic Process. (Figures behind ± are variances.)

Prediction	CNP	NP	AttnNP	DSVNP
Inter	-0.802±1E-6	-0.958±2E-5	**-1.149±8E-6**	-0.975±2E-5
Extra	**1.764±1E-1**	8.192±7E1	8.091±7E2	**4.203±9E0**

Further, we conduct curve fitting tasks in $\mathcal{SP}$. The $\mathcal{SP}$ initializes with a zero mean Gaussian Process $y^{(0)} \sim \mathcal{GP}(0, k(.,.))$ indexed in the interval $x \in [-2.0, 2.0]$, where the radial basis kernel $k(x, x') = \sigma^2 exp(-(x - x')^2/2l^2)$ is used with l-1 0.4 and σ 1.0. Then the transformation is performed to yield $y = \sin(y^{(0)}(x) + x)$. The training process follows that in NP (Garnelo et al., 2018b).

Predicted results are visualized in the second and the third rows of Fig. (3). Note that CNP only predicts points out of context in default settings. More evidence is reported in Table (3), where 2000 realizations are independently sampled and predicted for both interpolation and extrapolation.

After several repetitive observations, we find that in terms of the interpolation accuracy, DSVNP works better than vanilla NP, but the improvement is not as significant as that in AttnNP, which is also verified in visualizations. All (C)NPs show higher uncertainties around index 0, where fewer context points are located, and variances are relatively close in other regions. For extrapolation results, since all models are trained in the dotted column lines restricted regions, it is tough to scale to regions out of training interval, and all negative log-likelihoods (NLLs) are higher. When many context points exist outside the interval, the learned context variable may deteriorate predictions for all (C)NPs, and observations confirm findings in (Gordon et al., 2020b). Interestingly, DSVNP tends to overestimate uncertainties out of the training interval but predicted extrapolation results mostly fall into the one σ confident region, and this property is similar to CNP. On the other hand, vanilla NP and AttnNP sometimes underestimate the uncertainty.

3.5.2 *System Identification on Physics Engines*

Capturing dynamics in systems is crucial in control-related problems, and we extend synthetic experiments on a classical simulator, Cart-Pole systems, which is detailed in (Gal et al., 2016). As shown in Fig. (4), the original intention is to conduct actions to reach the goal with the end of a pole, but here we focus on dynamics, and the state is a vector of the location, the angle, and their first-order derivatives. Specifically, the aim is to forecast the transited state $[x_c, \theta, x_c', \theta']$ in time step $t+1$ based on the input as a state action pair $[x_c, \theta, x_c', \theta', a]$ in time step t.

To generate a variety of trajectories under a random policy for this experiment, the mass m_c and the ground friction coefficient f_c are varied in the discrete choices $m_c \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$ and $f_c \in \{0.06, 0.08, 0.1, 0.12\}$. Each pair of $[m_c, f_c]$ values specifies a dynamics environment, and we formulate all pairs of $m_c \in \{0.3, 0.5, 0.7\}$ and $f_c \in \{0.08, 0.12\}$ as training environments with the rest 16 pairs of configurations as the testing environments.

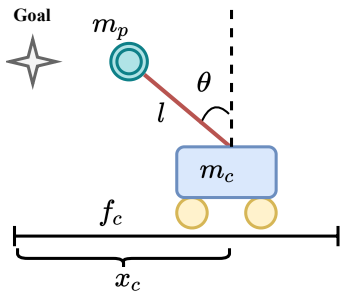


Figure 4: Cart-Pole Dynamical Systems.The cart and the pole are with masses m_c and m_p, and the length of the pole is l. And the configuration of the simulator is up to parameters of the cart-pole mass and the ground friction coefficient here with other hyper-parameters fixed in this experiment.

Table 4: Predictive Negative Log-Likelihoods and Mean Square Errors on Cart-Pole State Transition Testing Dataset. (Figures behind ± are variances.)

Metrics	CNP	NP	AttnNP	DSVNP
NLL	-2.014±9E-4	-1.537±1E-3	-1.821±7E-3	**-2.145±9E-4**
MSE	0.096±3E-4	0.074±2E-4	0.067±1E-4	**0.036±2.1E-5**

For each configuration of the simulator, including training and testing environments, we sample 400 trajectories of the horizon as 10 steps using a random controller. Please refer to Appendix (A.5) for more details. The training process follows Algorithm (1) with the maximum number of context points as 100. During the testing process, 100 state transition pairs are randomly selected for each environment configuration, working as the maximum context points to identify the dynamics configuration.

Furthermore, the collected results are reported in Table (4), where prediction performance on 5600 trajectories from 14 configurations is revealed. As can be seen, the negative log-likelihood values are not consistent with those of mean square errors, and DSVNP shows both better uncertainty quantification with lowest NLLs and approximation errors in MSEs. AttnNP improves NP in both metrics, while CNP shows relatively better NLLs, but the approximation error is a bit higher than others.

3.5.3 *Multi-Output Regression on Real-world Dataset*

Further, more complicated scenarios are considered when the regression task relates to multiple outputs. As investigated in (Moreno-Muñoz et al., 2018; Bonilla et al., 2008), distributions of output variables are implicit, which means no explicit distributions are appropriate to be used in parameterizing the output. We evaluate the performance of all models on dataset, including SARCOS [2], Water Quality (WQ) (Džeroski et al., 2000) and SCM20d (Spyromitros-Xioufis et al., 2016). Details about these datasets and neural architectures for all models are included in Appendix (A.5).

2 http://www.gaussianprocess.org/gpml/data/

Table 5: Predictive MSEs on Multi-Output Dataset. CNP's results are for target points. *D* records (input,output) dimensions. MC-Dropout runs 50 stochastic forward propagation and average results for prediction in each data point. (Figures behind ± are variances.)

Dataset	*D*	MC-Dropout	CNP	NP	AttnNP	DSVNP
Sarcos	(21,7)	1.215±3E-3	1.437±2.9E-2	1.285±1.2E-1	1.362±8.4E-2	**0.839±1.5E-2**
WQ	(16,14)	0.007±9.6E-8	0.015±2.4E-5	0.007±5.2E-6	0.01±8.5E-6	**0.006±1.6E-6**
SCM20d	(61,16)	0.017±2.4E-7	0.037±6.7E-5	0.015±7.1E-8	0.015±8.1E-7	**0.007±2.3E-7**

Furthermore, Monte-Carlo Dropout is included for comparison. Similar to NP (Garnelo et al., 2018b), the variance parameter is not learned, and the objective in optimization is pointwise mean square errors (MSEs) after averaging all dimensions in the output. Each dataset is randomly split into 2-folds as training and testing sets. The training procedure in (C)NPs follows that in Algorithm (1), and some context points are randomly selected in batch samples. We randomly select 30 instances in the testing stage as the context and then perform predictions with (C)NPs. The weights of data likelihood and KL divergence terms in models are not tuned here.

During training, ELBOs in NP-related models are optimized, while MSEs are used as an evaluation metric in testing (Dezfouli and Bonilla, 2015)[3]. The predictive results on the testing dataset are reported in Table (5). All MSEs are averaged after 10 independent experiments. We observe DSVNP outperforms other models, and deterministic context information in CNP hardly increases performance. Compared with NP models, MC-NN is relatively satisfying on Sarcos and WQ, and AttnNP works not well in these cases. A potential reason can be that deterministic context embedding with dot product attention is less predictive for output with multiple dimensions. At the same time, the role of the local latent variable in DSVNP not only bridges the gap between input and output but also extracts some correlation information among variables in outputs. The attention mechanism is more suitable for extracting local information when the output dimension is lower in synthetic experiments.

3.5.4 *Classification with Uncertainty Quantification*

Here image classification is performed with NP models and MC-Dropout, and out of distribution (o.o.d.) detection is chosen to measure the goodness of uncertainty quantification. We train models on MNIST and CIFAR10, and the dimensions for latent variables are 64 on MNIST and 128 on CIFAR10. The training process for NP related models follows Algorithm (1), with the number of context images randomly selected in each batch update.

We randomly select 100 instances from the domain dataset for the testing process as the context for (C)NP models. The commonly used measure for uncertainty in K-class o.o.d. detection is entropy (Lakshminarayanan et al., 2017), $\mathbb{H}[y^*|x^*] =$

3 Directly optimizing Gaussian log-likelihoods does harm to performance based on experimental results.

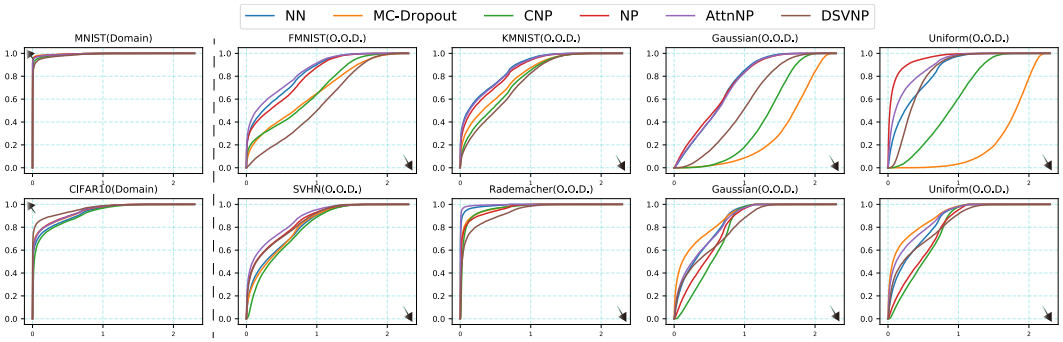


Figure 5: Cumulative Distribution Functions of Entropies in O.O.D. Detection Tasks. Values on X-axis are example entropies ranging from 0 to 2.3, and the y-axis records cumulative probabilities. The first row corresponds to the predictive result with models trained on MNIST, while the second is with models trained on CIFAR10. NN means the baseline neural network without a dropout layer. Curves in CDFs closer to triangular arrows reveal better uncertainty quantification.

$-\Sigma_{c=1}^{K} p_w(y_c^*|x^*;\mathcal{D}_{tr}) \ln p_w(y_c^*|x^*;\mathcal{D}_{tr})$, where data point (x^*, y^*) comes from either domain test dataset $\mathcal{D}_{te}$ or o.o.d dataset $\mathcal{D}_{ood}$.

For classification performance with NP related models, we observe the difference is extremely tiny on MNIST with all accuracies around 99%, while on CIFAR10, DSVNP beats all baselines with the highest accuracy 86.3% and lowest in-distribution entropies in Table (6). The involvement of a deterministic path does not improve much, and in contrast, MC-Dropout and CNP achieve intermediate performance. A possible cause can be that implicit kernel information captured by the attention network in images is imprecise.

The cumulative distributions of predictive entropies are reported in Fig. (5). For models trained on MNIST, we observe no significant difference in the domain dataset, but DSVNP achieves the best results on FMNIST/KMNIST, and MC-Dropout performs superior on the Uniform/Gaussian noise dataset. Interestingly, AttnNP tends to underestimate uncertainty on FMNIST/KMNIST, and the measure is close to the neural network without dropout. Those trained on CIFAR10 differ from observations in the second row of Fig. (5). It can be noticed that DSVNP shows the lowest uncertainty on the domain dataset (CIFAR10) and medium uncertainty on SVHN/Gaussian/Uniform Dataset. MC-Dropout and AttnNP seem not to work so well overall, but CNP well measures uncertainty on Gaussian/Uniform dataset. Results again verify SVHN as a tough dataset for the task (Nalisnick et al., 2019). Also, note that entropy distributions on Rademacher Dataset are akin to that on the domain dataset, which means the Rademacher noise is riskier for CIFAR10 classification, and DSVNP is a better choice to avoid such adversarial attack in this case.

That evidence shows that the deterministic path in AttnNP does not boost classification performance on the domain dataset but weakens the ability of o.o.d. detection mostly, while local latent variables in DSVNP improve both performances. Maybe deterministic local latent variables require more practical attention information, but here only dot-product attention information is included. As a comparison, the local latent variable

Table 6: Tested Entropies of Logit Probability on Classification Dataset. For rows of MNIST and CIFAR10, the second figures in columns are classification accuracies. Both MC-Dropout and DSVNP are averaged with 100 Monte Carlo samples.

	NN	MC-Dropout	CNP	NP	AttnNP	DSVNP
MNIST	0.011/0.990	**0.009/0.993**	0.019/0.993	0.010/0.991	0.012/0.989	0.027/0.990
FMNIST	0.385	0.735	0.711	0.434	0.337	**0.956**
KMNIST	0.282	0.438	0.497	0.322	0.294	**0.545**
Gaussian	0.601	**1.623**	1.313	0.588	0.611	0.966
Uniform	0.330	**1.739**	0.862	0.094	0.220	0.375
CIFAR10	0.151/0.768	0.125/0.838	0.177/0.834	0.124/0.792	0.124/0.795	**0.081/0.863**
SVHN	0.402	0.407	**0.459**	0.315	0.269	0.326
Rademacher	0.021	0.062	0.079	0.078	0.010	**0.146**
Gaussian	0.351	0.266	**0.523**	0.451	0.349	0.444
Uniform	0.334	0.217	**0.499**	0.463	0.261	0.374

in DSVNP captures some target-specific information during the training process and improves detection performance.

3.6 CONCLUSION & DISCUSSION

Technical Discussions. This chapter presents a novel exchangeable stochastic process as DSVNP, formulated as a latent variable model. DSVNP integrates latent variables hierarchically and improves the expressiveness of the vanilla NP model. Experiments on high-dimensional tasks demonstrate better capability in prediction and uncertainty quantification.

Existing Limitations. DSVNP achieves satisfying performance in the benchmark mentioned above, but it suffers from the risk of underfitting when the output dimension is low. This is because we apply the inference way of C-VAEs (Sohn et al., 2015) to local latent variables with the output in the approximate posterior. In this case, more weights are assigned to KL divergence penalties of local latent variables.

Future Extensions. Since DSVNP mainly concentrates on latent variables and associated inference methods, future directions can enhance latent variables' representation, such as using more flexible equivariant transformations over the context or the dedicated selection of proper context points.

4

MIXTURE OF EXPERTS STRUCTURES FOR FUNCTIONAL REPRESENTATION

In the previous chapter (3), we examined the usefulness of the Bayesian hierarchical inductive bias in neural processes for high dimensional learning problems. This chapter considers the scenario when learning datasets originate from a mixture of function distributions. To solve the problem, we incorporate the mixture of expert inductive bias into neural processes. The resulting model can effectively represent functions from mixture distributions and achieve state-of-art performance in experiments.

4.1 INTRODUCTION

Humans can naturally adapt to new environments after developing related skills, which relies on the excellent abstraction of environments. Similarly, *meta learning* or *learning to learn* tries to leverage past experiences, and with the help of the incorporated meta learned knowledge, a new skill can be mastered rapidly with a few instances.

During the past decade, an increasing number of methods have emerged in meta learning domains. In this paper, we concentrate on a particular branch of meta learning methods, referred to as context-based meta learning (Garnelo et al., 2018b,a). A representative one is a neural process (NP) (Garnelo et al., 2018b), which was initially proposed to approximate Gaussian processes with lower computational cost. The core purpose of NPs is to learn meta-representations (Gondal et al., 2021), which encode context points into latent variables and represent the task in a functional form. In comparison to gradient-based meta learning algorithms, *e.g.* model-agnostic meta learning (MAML) (Finn et al., 2017), the NP directly learns a functional representation and does not require additional gradient updates in fast adaptation.

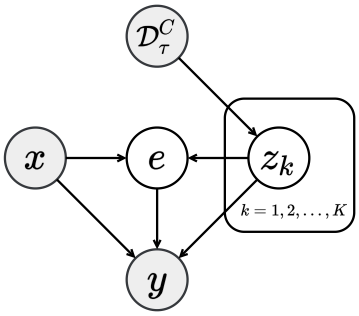


Figure 6: Generative Process of MoE-NPs. Here $\mathcal{D}_\tau^C$ refers to dataset of context points in the paper. $\{z_k\}_{k=1}^K$ are a set of expert latent variables and e is an assignment latent variable. Observed variables are grey in circles with latent variables white.

Research Motivations. Fundamentally, vanilla NPs employ global Gaussian latent variables to specify different tasks. This setting raises several concerns in some scenarios. (i) When observations originate from a mixture of stochastic processes (Rasmussen and Ghahramani, 2002), a single Gaussian latent variable is faced with deficiencies in

formulating complex functional forms. (ii) In context-based meta reinforcement learning, the uncertainty of value functions revealed from latent variables encourages effective exploration in environments (Rakelly et al., 2019). However, when multiple variates govern tasks, *e.g.* velocities, masses, or goals in Mujoco robots (Todorov et al., 2012), the use of a unimodal Gaussian latent variable restricts the flexibility of randomized value functions (Osband et al., 2016b), leading to sub-optimality in performance.

Developed Methods. Instead of using a global latent variable in modeling, we employ multiple latent variables to induce a mixture of expert NPs to specify diverse functional priors. Meanwhile, the discrete latent variables as assignment variables are introduced to establish connections between a single data point and expert NPs. We utilize variational inference to formulate the evidence lower bound to optimize this model with hybrid types of latent variables. Additionally, special modules are designed to accommodate few-shot supervised learning and meta reinforcement learning tasks.

Our primary contributions are two-fold in this chapter:

- We introduce a new exchangeable stochastic process, referred to as MoE-NPs, to enrich the family of NPs. Our model inherits the advantages of both mixtures of expert models and NPs, enabling multi-modal meta representations for functions.
- We specify inference modules in MoE-NPs for few shot supervised learning and meta reinforcement learning tasks. Extensive meta learning experiments show that MoE-NPs can achieve competitive performance compared to most existing methods.

4.2 RELATED WORK

Meta Learning. Meta learning is a paradigm to enable fast learning (fast adaptation to new tasks) via slow learning (meta training in the distribution of tasks). There exist several branches of meta learning algorithms. Gradient-based meta learning algorithms, *e.g.* MAML (Finn et al., 2017) and its variants (Zintgraf et al., 2019; Rajeswaran et al., 2019; Finn et al., 2018), perform gradient updates over model parameters to achieve fast adaptation with a few instances. Metrics-based meta learning algorithms try to learn representations of tasks in distance space, and models *e.g.* prototypical networks (Snell et al., 2017; Allen et al., 2019) are popular in computer vision domains. As for context-based meta learning methods of our interest, latent variable models, *e.g.* NPs (Garnelo et al., 2018b), are designed to learn task representations in a functional space. This family does not require gradient updates in fast adaptation.

Neural Processes Family. Apart from vanilla NPs or CNPs (Garnelo et al., 2018a,b), other variants are developed, and these are built on various inductive biases. To address underfitting issues, attention networks (Kim et al., 2019a, 2021) are introduced to augment NPs. (Foong et al., 2020; Gordon et al., 2020a) improve the generalization capability of NPs with help of convolutional operations. To learn distributions of group equivariant functions, (Kawano et al., 2020) has proposed EquiCNP. Similarly, SteerC-NPs also incorporate equivariance to approximate stochastic fields (Holderrieth et al.,

2021). Our work is to get NPs married with Mixture of Experts (MoEs) models (Xu et al., 1995; Waterhouse et al., 1996), which model datasets with a collection of expert NPs. We provide more technical summary of the NP family together with other probabilistic meta learning methods (Gordon et al., 2018; Iakovleva et al., 2020; Requeima et al., 2019; Sun et al., 2021) in the Appendix (B.5).

4.3 PRELIMINARIES

Notations. The paradigm of *meta learning* is considered in the distribution of tasks $p(\mathcal{T})$, and a task sampled from $p(\mathcal{T})$ is denoted by τ in this paper. The form of a task depends on applications. For example, a task of our interest in regressions can be a function f to fit, which is a realization from unknown stochastic processes (Rasmussen, 2003).

Let $\mathcal{D}_\tau^C$ refer to a set of context points used to specify the underlying task, and $\mathcal{D}_\tau^T = [x_T, y_T]$ is a set of target points to predict, *e.g.* $f(x_T) = y_T$. In the context-based meta learning with latent variables, we write the probabilistic dependencies in distributions of target functions as follows,

$$p(f(x_T)|\mathcal{D}_\tau^C, x_T) = \int p(f(x_T)|z, x_T)p(z|\mathcal{D}_\tau^C)dz \tag{4.1}$$

where the functional prior $p(z|\mathcal{D}_\tau^C)$ is injected in modeling via latent variables z.

Neural Processes. The family of NPs (Garnelo et al., 2018b) belongs to exchangeable stochastic processes (Ross et al., 1996). A generative process is written as Eq. (4.2) with a global Gaussian latent variable z placed in Eq. (4.1),

$$\rho_{x_{1:N}}(y_{1:N}) = \int p(z) \prod_{i=1}^{N} \mathcal{N}(y_i|f_\theta(x_i, z), \sigma_i^2)dz \tag{4.2}$$

where f_θ is a mean function and σ_i^2 is the corresponding variance. In our settings, we treat the conditional neural process (Garnelo et al., 2018a) as a special case in NPs, when the distribution of z is collapsed into a Dirac delta distribution $p(z) = \delta(z - \hat{z})$ with $\hat{z}$ a fixed real valued vector.

4.3.1 *Few-Shot Supervised Learning*

In the context-based meta learning, we formulate the few-shot supervised learning objective within the expected risk minimization principle as follows.

$$\min_{\Theta} \mathbb{E}_{\tau \sim p(\mathcal{T})}\Big[\mathcal{L}(\mathcal{D}_\tau^T; \mathcal{D}_\tau^C, \Theta)\Big] \tag{4.3}$$

The risk function $\mathcal{L}$, *e.g.* negative log-likelihoods, measures performance of meta learning structure on the task-specific dataset $\mathcal{D}_\tau^C$ and $\mathcal{D}_\tau^T$, and Θ means parameters of common knowledge shared across tasks and parameters for fast adaptation (*e.g*, Θ denotes parameters of the encoder ϕ and decoder θ, and $\mathcal{L}$ is the approximate objective in NPs).

With the set of context points $\mathcal{D}_\tau^C = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and the target points $\mathcal{D}_\tau^T$, the posterior of a global latent variable z in Eq. (4.1) is approximated with a variational distribution $q_\phi(z|\mathcal{D}_\tau^T)$ and an evidence lower bound (ELBO) is derived to optimize in practice. A general meta training objective of NPs in few-shot supervised learning is $\mathcal{L}(\theta, \phi)$ in Eq. (4.4), where $p_\theta(\mathcal{D}_\tau^T|z) = \prod_{i=1}^n p_\theta(y_i|[x_i, z])$.

$$\mathbb{E}_\tau\left[\ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C)\right] \geq \mathbb{E}_\tau\left[\mathbb{E}_{q_\phi(z|\mathcal{D}_\tau^T)}[\ln p_\theta(\mathcal{D}_\tau^T|z)] - D_{KL}[q_\phi(z|\mathcal{D}_\tau^T) \parallel q_\phi(z|\mathcal{D}_\tau^C)]\right] \quad (4.4)$$

4.3.2 *Meta Reinforcement Learning*

For the context-based meta reinforcement learning, the context points $\mathcal{D}_\tau^C$ are a set of random transition samples from an environment as $\mathcal{D}_\tau^C = \{(s_1, a_1, s_2, r_1), \ldots, (s_H, a_H, s_{H+1}, r_H)\}$, where r_t is the one-step reward after performing action a_t at state s_t. Here $\mathcal{D}_\tau^C$ plays a role in task inference (Humplik et al., 2019) to obtain the information bottleneck $q_\phi(z|\mathcal{D}_\tau^C)$ and $\mathcal{D}_\tau^T$ is the dataset of state action values to fit.

For example, in an off-policy meta reinforcement learning algorithm, *e.g.* PEARL (Rakelly et al., 2019) or FCRL (Gondal et al., 2021), the general optimization objective consists of two parts: (i) to approximate distributions of task-specific optimal value functions in Eq. (4.5), where Q_θ is optimal Q-value with the state value $\hat{V}$ (ii) to maximize the cumulative rewards $\mathbb{E}_\tau\left[\mathbb{E}_{q_\phi(z|\mathcal{D}_\tau^C)}[\mathcal{R}(\tau, z; \varphi)]\right]$, where $\mathcal{R}$ is the expected cumulative rewards in the environment τ given policies $\pi_\varphi(a|[s, z])$.

$$\begin{aligned}\mathcal{L}(\theta, \phi) = \mathbb{E}_\tau \mathbb{E}_{\substack{(s,a,s',r)\sim\mathcal{D}_\tau^T \\ z\sim q_\phi(z|\mathcal{D}_\tau^C)}}&[Q_\theta([s, z], a) - (r + \hat{V}([s', z]))]^2 \\ &+\beta\mathbb{E}_\tau\left[D_{KL}[q_\phi(z|\mathcal{D}_\tau^C) \parallel p(z)]\right]\end{aligned} \quad (4.5)$$

Different from the few-shot supervised learning, here, the context points are not part of the fitting dataset, which means $\mathcal{D}_\tau^C \not\subset \mathcal{D}_\tau^T$. As implemented in (Rakelly et al., 2019), the prior distribution $p(z)$ is typically selected as a fixed one, *e.g.* $\mathcal{N}(0, I)$. The induced distribution of task-specific value functions $p(Q_\theta([s, z], a))$ enables posterior sampling (Russo and Van Roy, 2014) in meta learning scenarios, which brings additional benefits of exploration for continuous control problems.

4.4 METHODS

This section presents our developed MoE-NPs and connects them to the hierarchical Bayes framework. Then approximate objectives are derived, and stochastic gradient variational Bayes (Kingma and Welling, 2013) is used to optimize the developed model. Finally, specialized neural modules are described for MoE-NPs application to different

meta learning tasks. Detailed computational diagrams in training and testing are attached in Appendix (B.2). For simplicity, we derive equations *w.r.t.* a task τ in the following section, but a batch of tasks are considered in training in implementation.

4.4.1 *Mixture of Expert Neural Processes*

Vanilla NPs often suffer underfitting in experiments (Garnelo et al., 2018b; Kim et al., 2019a). This can be attributed to expressiveness bottlenecks when employing a global latent variable in learning functional priors of tasks from unknown distributions (Dilokthanakul et al., 2016).

To alleviate mentioned deficiencies, we make two modifications for the NP family. (i) Multiple functional priors are encoded in modeling with the help of K expert latent variables, which can capture statistical traits, *e.g.* distributional multi-modality, in data points. This setting is also an extension of Mixture of Experts (MoEs) models (Xu et al., 1995; Shazeer et al., 2016; Yuksel et al., 2012; Pei et al., 2020) to meta learning scenarios. (ii) Like the gating mechanism in (Rasmussen and Ghahramani, 2002), assignment latent variables are included in modeling to select functional forms for each data point in prediction. The resulting MoE-NPs can learn more expressive functional priors and exhibit the approximation power for local properties of the dataset.

Generative Process

As displayed in Fig. (6), the graphical model involves two types of latent variables respectively the continuous expert latent variables $z_{1:K}$ and the discrete assignment latent variable e. Further, we can translate the generative process into equations as follows,

$$\rho_{x_{1:N}}(y_{1:N}) = \int \prod_{k=1}^{K} p(z_k) \cdot \prod_{i=1}^{N} \left[\sum_{k=1}^{K} p(y_i|x_i, z_{1:K}, e_k = 1) p(e_k = 1|x_i, z_{1:K}) \right] dz_{1:K} \tag{4.6}$$

where the sampled assignment variable e is in the form of K-dimensional one-hot encoding, and $e_k = 1$ in Eq. (4.6) indicates the k-th expert z_k is selected from $z_{1:K}$ for prediction. A more detailed probabilistic generative process can also be found in Appendix (B.4.1). In this way, our developed model constitutes an *exchangeable stochastic process*. And we demonstrate this claim with help of Kolmogorov Extension Theorem (Bhattacharya and Waymire, 2009) in Appendix (B.4.2).

Link to Hierarchical Bayes

Note that latent variables in Eq. (4.6) are of hybrid types. K functional priors are incorporated in expert latent variables $z_{1:K}$, while the assignment latent variable e is input dependent. The dependencies between $z_{1:K}$ and e are reflected in modeling, and this connects our work to Hierarchical Bayes (Lawrence and Platt, 2004; Daumé III, 2009) in a latent variable sense. Also, when only one expert latent variable is used here, the hierarchical model degenerates to the vanilla (C)NPs (Garnelo et al., 2018b,a).

4.4.2 *Scalable Training & Prediction*

Inference Process

Given a task τ, due to existence of unknown latent variables, it is intractable to perform exact inference *w.r.t.* $p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C)$. As an alternative, we apply variational inference to our developed model. Here we use $[x, y]$ to denote a single data point from a set of target points $\mathcal{D}_\tau^T$.

$$\ln p(y|x, \mathcal{D}_\tau^C) \geq \mathbb{E}_{q_{\phi_1}}\left[\mathbb{E}_{q_{\phi_{2,1}}}\left[\ln p_\theta(y|x, z_{1:K}, e)\right] - D_{KL}[q_{\phi_{2,1}}(e|x, y, z_{1:K}) \parallel p_{\phi_{2,2}}(e|x, z_{1:K})]\right] - \sum_{k=1}^{K} D_{KL}[q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^T) \parallel q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^C)] = -\mathcal{L}(\theta, \phi_1, \phi_2) \tag{4.7}$$

An example for the k-th expert latent variable z_k can be in the form of a Gaussian distribution $\mathcal{N}(z_k; \mu_k, \Sigma_k)$. And in meta training processes, the assignment variable e is assumed to be drawn from a categorical distribution $\texttt{Cat}(e; K, \alpha(x, y, z_{1:K}))$ with parameters $\alpha(x, y, z_{1:K})$.

The existence of discrete latent variables e makes it tough to optimize using traditional methods. This is because either sampling algorithms or expectation maximization algorithms are computationally intensive when utilized here (we have discussed this point in Appendix (B.6)) for expert latent variables.

To reduce the computational cost, we again utilize variational inference and the decoder directly formulates the output as a mixture of log-likelihoods $\mathbb{E}_{q_{\phi_{2,1}}}[\ln p_\theta(y|x, z_{1:K}, e)] = \sum_{k=1}^{K} \alpha_k(x, y, z_{1:K}; \phi_{2,1}) \ln p_\theta(y|x, z_k)$.

This results in a general ELBO as Eq. (4.7) for few-shot supervised learning in meta training, where q_{ϕ_1} denotes a collection of K independent variational distribution $\{q_{\phi_{1,1}}, \ldots, q_{\phi_{1,K}}\}$. $q_{\phi_{2,1}}$ and $p_{\phi_{2,2}}$ respectively define the variational posterior and prior for assignment latent variables. Please refer to Appendix (B.2)/(B.6) for definitions and more detailed derivations.

Monte Carlo Estimates & Predictions

Meta-training processes consider a batch of tasks to optimize in iterations, and we apply Monte Carlo methods to the obtained negative ELBO $\mathcal{L}(\theta, \phi_1, \phi_2)$ as follows.

$$\begin{aligned}\mathcal{L}_{MC}(\theta, \phi_1, \phi_2) = &-\frac{1}{NB}\sum_{b=1}^{B}\sum_{i=1}^{N}\left[\sum_{k=1}^{K}\alpha_k^{(b)} \ln p(y_i^{(b)}|x_i^{(b)}, z_k^{(b)})\right] \\ &+\frac{1}{NB}\sum_{b=1}^{B}\sum_{i=1}^{N} D_{KL}[q_{\phi_{2,1}}(e_i^{(b)}|x_i^{(b)}, y_i^{(b)}, z_{1:K}^{(b)}) \parallel p_{\phi_{2,2}}(e_i^{(b)}|x^{(b)}, z_{1:K}^{(b)})] \\ &+\frac{1}{NB}\sum_{b=1}^{B}\sum_{k=1}^{K} D_{KL}[q_{\phi_{1,k}}(z_k^{(b)}|\mathcal{D}_b^T) \parallel q_{\phi_{1,k}}(z_k^{(b)}|\mathcal{D}_b^C)]\end{aligned} \tag{4.8}$$

With the number of tasks B and the number of data points N in mini-batches, the Monte Carlo estimate with one stochastic forward pass is Eq. (4.8) for meta training objectives.

Like that in NPs (Garnelo et al., 2018b), we derive the predictive distribution as Eq. (4.9) with one stochastic forward pass and parameters of discrete latent variables $p_{\phi_{2,2}}(e_k = 1|x_*, z_{1:K}) = \alpha_k(x, z_{1:K}; \phi_{2,2})$.

$$p(y_*|x_*, \mathcal{D}_\tau^C) \approx \sum_{k=1}^{K} \alpha_k(x, z_{1:K}; \phi_{2,2}) p_\theta(y|x_*, z_k) \quad \text{with } z_{1:K} \sim q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^C) \tag{4.9}$$

And the point estimate in prediction $\mathbb{E}[Y|X = x, \mathcal{D}_\tau^C]$ can also be obtained in Appendix (B.6.4).

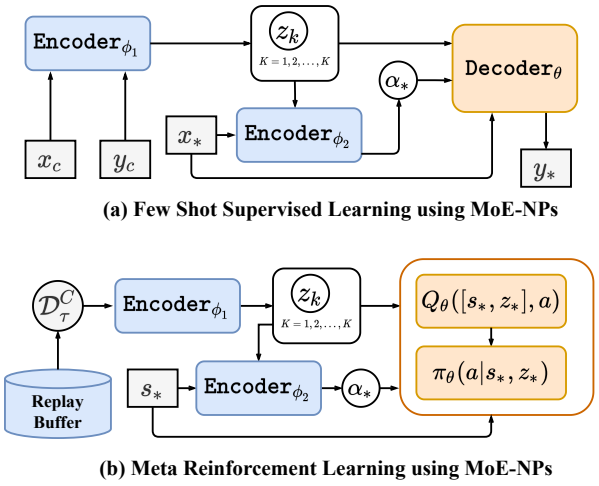


Figure 7: Computational Diagram of MoE-NPs in Meta Testing. **In (a):** The context variables are $\mathcal{D}_\tau^C = [x_C, y_C]$, and the expert latent variables $z_{1:K}$ are approximated with neural networks. For discrete assignment latent variables e_*, we learn parameters of categorical distributions α_* with neural networks. **In (b):** The context variables $\mathcal{D}_\tau^C$ are sampled transitions from a memory buffer. During policy search, the selected expert latent variable z_* is a context variable in both Actor and Critic networks.

4.4.3 *Module Details for Meta Learning*

Inference Modules in MoE-NPs

The equations define a framework that can be implemented in different meta learning tasks. Two examples are given in Fig. (7). Note that two types of latent variables are involved in modeling, we need different structures of encoders for latent variables. Inference Modules are required for them to satisfy different conditions. In variational inference, the distribution parameters of these latent variables are approximated with the output of these specialized encoders.

Inference Modules for Continuous Latent Variables. For neural networks to parameterize the encoder of continuous latent variables q_{ϕ_1}, we use the same architectures in (C)NPs (Garnelo et al., 2018b,a), which are permutation invariant to the order of context points $[x_C, y_C] = \{[x_1, y_1], \ldots, [x_N, y_N]\}$. That is, for any permutation operator σ over the set of context points, the neural network ($\texttt{NN}$) parameters of an output distribution for each expert z_k should satisfy $[\mu_k, \Sigma_k] = \texttt{NN}_{\phi_{1,k}}([x_{\sigma(1:N)}, y_{\sigma(1:N)}])$.

$$r_{k,i} = h_{\phi_{1,k}}([x_i, y_i]),\ r_k = \bigoplus_{i=1}^{N} r_{k,i},\ [\mu_k, \Sigma_k] = g_{\phi_{1,k}}(r_k) \tag{4.10}$$

Eq. (4.10) is an example, where h is the embedding function, $\bigoplus$ denotes a mean pooling operation, and g is part of encoder networks.

Inference Modules for Categorical Latent Variables. For neural networks to parameterize the encoder of discrete latent variables $q_{\phi_{2,1}}$ and $p_{\phi_{2,2}}$, we need the categorical distribution parameters α to be permutation equivariant (Finzi et al., 2021) with respect to the order of $z_{1:K}$. This means for any order permutation operation σ, the condition is satisfied as $[\alpha_{\sigma(1)}, \alpha_{\sigma(2)}, \ldots, \alpha_{\sigma(K)}] = \texttt{NN}_{\phi_{2,1}}(x, y, z_{\sigma(1:K)})$.

$$b_k = h_{\phi_{2,1}}(x, y, z_k)\ \forall k \in \{1, 2, \ldots, K\}, \quad [\alpha_1, \alpha_2, \ldots, \alpha_K] = \texttt{softmax}(b/t) \tag{4.11}$$

An example implementation for the variational posterior $\texttt{Cat}(e; K, \alpha(x, y, z_{1:K}))$ can be Eq. (4.11), where the vector of logits is $b = [b_1, b_2, \ldots, b_K]$ with t a temperature parameter. And this implementation applies to prior networks $\texttt{NN}_{\phi_{2,2}}(x, z_{\sigma(1:K)})$ to learn distribution parameters of $\texttt{Cat}(e; K, \alpha(x, z_{1:K}))$ in the same way.

Meta RL Modules in MoE-NPs.

When extending MoE-NPs to meta RL tasks, optimization objectives in Eq. (4.7) need to be modified for Actor-Critic methods, which are employed in our settings. Like that in PEARL (Rakelly et al., 2019) and FCRL (Gondal et al., 2021), the soft actor critic (SAC) algorithm (Haarnoja et al., 2018) is used to learn policies due to good sample efficiency.

Given a specific MDP τ, posterior distributions of optimal value functions are formulated via latent variables z in context-based meta RL. That is, $p(Q_\theta(s, a; \mathcal{M}))$ is approximated in the form $p(Q_\theta([s, z], a))$. The resulting objectives for the Actor and Critic functions are respectively in Eq. (4.12) and Eq. (4.13), where $\mathcal{Z}_\theta$ is a normalization factor.

$$\mathcal{L}_A^\tau = \mathbb{E}_{\substack{s\sim\mathcal{D}_\tau^T, a\sim\pi_\varphi \\ z\sim q_\phi}} \left[D_{KL} \left[\pi_\varphi(a|[s, z]) \parallel \frac{\exp\{(Q_\theta([s, z], a))\}}{\mathcal{Z}_\theta(s)} \right] \right] \tag{4.12}$$

The variational posterior $q_\phi(z|s, \mathcal{D}_\tau^C)$ in Eq. (4.13) is a state dependent distribution with $\hat{V}$ a state value function, and sampling processes refer to steps in Algorithm (4).

$$\mathcal{L}_C^\tau = \mathbb{E}_{\substack{(s,a,s',r)\sim\mathcal{D}_\tau^T \\ z,z'\sim q_\phi}} [Q_\theta([s,z],a) - (r + \hat{V}([s',z']))]^2 \tag{4.13}$$

A key difference from PEARL (Rakelly et al., 2019) lies in that several expert latent variables and assignment latent variables are involved in modeling. So we refer the Kullback–Leibler divergence term to Eq. (4.14) in MoE-NPs with coefficient β_0 and β_1.

$$\mathcal{L}_{KL}^\tau = \beta_1 \mathbb{E}_{\substack{(s,a,s',r)\sim\mathcal{D}_\tau^T \\ q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^C)}} [D_{KL}[q_{\phi_2}(e|s,z_{1:K}) \parallel p(e)]] + \beta_0 \sum_{k=1}^{K} D_{KL}[q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^C) \parallel p(z_k)] \tag{4.14}$$

The Monte Carlo estimates *w.r.t.* Eq. (4.12/4.13/4.14) are used in meta training, and Pseudo code to optimize these functions is listed in Algorithm (4).

Algorithm 2: MoE-NPs for Few-Shot Supervised Learning.

Input : Task distribution $p(\mathcal{T})$; Task batch size $\mathcal{B}$; Length of mini-batch instances N_{max}; Epochs m; Learning rates λ_1 and λ_2.

Output : Meta-trained parameters $\phi = [\phi_1, \phi_2]$ and θ.

Initialize parameters ϕ and θ;
for $i = 1$ ***to*** m **do**
  Sample a random value $N_C \sim U[1, N_{max}]$ as the number of context points;
  Sample mini-batch instances $\mathcal{D}$ to split dataset $\{(x_C, y_C, x_T, y_T)_{bs}\}_{bs=1}^B$;
  // generative process
  Sample expert latent variables $z_{1:K} \sim q_{\phi_1}(z_{1:K}|\mathcal{D}^T)$;
  Compute distribution parameters α with Eq. (4.11);
  Compute negative ELBOs $\mathcal{L}_{MC}(\theta,\phi)$ in Eq. (4.8);
  // amortized inference process
  $\phi \leftarrow \phi - \lambda_1 \nabla_\phi \mathcal{L}_{MC}(\theta,\phi)$ in Eq. (4.8);
  $\theta \leftarrow \theta - \lambda_2 \nabla_\theta \mathcal{L}_{MC}(\theta,\phi)$ in Eq. (4.8);
end

Algorithm 3: MoE-NPs for Few Shot Supervised Learning (Meta-Testing Phases).

Input : Task τ; Meta-trained $\phi = [\phi_1, \phi_{2,2}]$ and θ.
Output : Predictive distributions.

Initialize parameters ϕ and θ;
Set the number of context points N_C;
Split test dataset into the context/target $\{(x_C, y_C, x_T, y_T)_{bs}\}_{bs=1}^{B} \sim \mathcal{D}$;
// generative process
Sample expert latent variables of the mini-batch $z_{1:K} \sim q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^C)$;
if *discrete l.v.s for hard assignment* **then**
    Sample assignment latent variables of the mini-batch $e \sim p_{\phi_{2,2}}(e|x_T, z_{1:K})$;
    Output the distribution $p_\theta(y_T|x_T, z_{1:K}, e)$;
else
    Compute the distribution parameters α of assignment latent variables via Eq. (4.11);
    Output the predictive distribution $p_\theta(y_T|x_T, \mathcal{D}_\tau^C)$ via Eq. (4.9);
end

Algorithm 4: MoE-NPs for Meta RL.

Input : MDP distribution $p(\mathcal{T})$; Batch size of tasks $\mathcal{B}$; Training steps m; Learning rates λ_1, λ_2 and λ_3.

Output : Meta-trained parameters ϕ, θ and φ.

Initialize parameters ϕ, θ, φ and replay buffer$\{\mathcal{M}_\tau^C\}^{\mathcal{B}}$;
while *Meta-Training not Completed* **do**
  Sample a batch of tasks $\{\tau\}^{\mathcal{B}} \sim p(\mathcal{T})$;
  // collect context transitions
  for *each* $\tau \in \{\tau\}^{\mathcal{B}}$ **do**
    Initialize the context $\mathcal{D}_\tau^C = \{\}$;
    Execute Algorithm (5) in Appendix
    to update $\mathcal{D}_\tau^C$
  end
  // actor critic learning in batchs
  for $i = 1$ ***to*** m **do**
    for *each* $\tau \in \{\tau\}^{\mathcal{B}}$ **do**
      Sample context points $\mathcal{D}_\tau^C \sim \mathcal{S}_c(\mathcal{M}_\tau^C)$
      & batch of transitions $b_\tau \sim \mathcal{M}_\tau$;
      Sample $z_{1:K} \sim q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^C)$;
      for *each* $s \in b_\tau$ **do**
        Sample $e \sim q_{\phi_2}(e|z_{1:K}, s)$ to select z
        and augment the state as $[s, z] \in b_\tau$;
      end
      // run forward propagation
      $\mathcal{L}_A^\tau = \mathcal{L}_A^\tau(b_\tau)$ in Eq. (4.12);
      $\mathcal{L}_C^\tau = \mathcal{L}_C^\tau(b_\tau)$ in Eq. (4.13);
      $\mathcal{L}_{KL}^\tau = \mathcal{L}_{KL}^\tau(\mathcal{D}_\tau^C, b_\tau)$ in Eq. (4.14)
    end
    // run back propagation
    $\phi \leftarrow \phi - \lambda_1 \nabla_\phi \sum_\tau (\mathcal{L}_C^\tau + \mathcal{L}_{KL}^\tau)$ in Eq. (4.13/4.14);
    $\varphi \leftarrow \varphi - \lambda_2 \nabla_\varphi \sum_\tau \mathcal{L}_A^\tau$ in Eq. (4.12);
    $\theta \leftarrow \theta - \lambda_3 \nabla_\theta \sum_\tau \mathcal{L}_C^\tau$ in Eq. (4.13);
  end
end

4.5 EXPERIMENTS

4.5.1 *General Setup*

The implementation of MoE-NPs in meta training can be found in Algorithms (2)/(4), and also please refer to Algorithms (3)/(5) for the corresponding meta-testing processes. We leave the details of experimental implementations (*e.g.* parameters, neural architectures, corresponding PyTorch modules and example codes) in Appendix (B.7).

Algorithm 5: MoE-NPs for Meta RL (Meta-Testing Phases).

Input : MDP distribution $p(\mathcal{T})$; meta-trained parameters ϕ, θ.
Output : Cumulative rewards.

Sample a test task $\tau \sim p(\mathcal{T})$;
Initialize parameters ϕ, θ, φ and replay buffer $\mathcal{M}_\tau^C$;
// collect transitions for memory buffers
Initialize the context $\mathcal{D}_\tau^C = \{\}$;
for $k = 1, 2, \ldots, K$ **do**
    Sample $z_{1:K} \sim q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^C)$;
    for *state s of each time step* **do**
        Sample $e \sim q_{\phi_2}(e|s, z_{1:K})$;
        Gather data from $\pi_\varphi(a|[s, z_{1:K}, e])$ to update $\mathcal{M}_\tau^C$;
        Update $\mathcal{D}_\tau^C = \{(s_j, a_j, s'_j, r_j)\}_{j=1}^N \sim \mathcal{M}_\tau^C$;
    end
end

Baselines for Learning Tasks. Apart from MoE-NPs, methods involved in comparisons are context-based methods such as CNPs (Garnelo et al., 2018a), NPs (Garnelo et al., 2018b) and FCRL (Gondal et al., 2021), and gradient-based methods such as MAML (Finn et al., 2017) and CAVIA (Zintgraf et al., 2019). For FCRL, contrastive terms from SimCLR (Chen et al., 2020) are included in the objective. In meta RL, the modified NP model corresponds to PEARL (Rakelly et al., 2019). Meanwhile, in Appendix (B.8), we include additional experimental results compared with other NPs models augmented by attentive modules (Kim et al., 2019a) or convolutional modules (Gordon et al., 2020a).

4.5.2 *Illustration in Toy Regression*

We visually show the effects of stochastic function fitting and quantified uncertainty in the toy dataset to see the different roles of latent variables. Our goal is to discover potential components of distributions from limited observed data points.

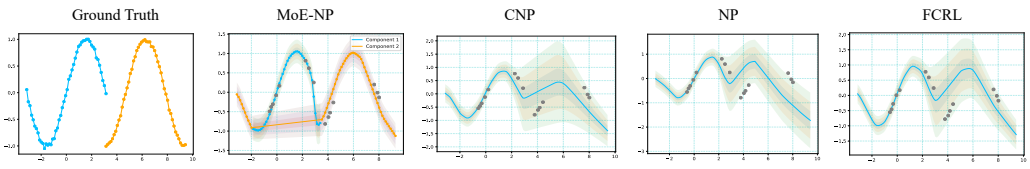


Figure 8: The Ground Truth and Predictive Distributions of Curves using NP related Models. The gray dots around curves are the context points. The shaded regions correspond to 3x standard deviations. In MoE-NPs, two components of the sampled mixture curve in blue and orange can be identified via assignment latent variables with more than 85% accuracy.

The learning data points are sampled in x-domain $[-\pi, 3\pi]$ and merged from a mixture of randomized functions $f_1(x) = \sin(x) + \epsilon_1$ and $f_2(x) = \cos(x) + \epsilon_2$ with equal probability for mixture components, where $\epsilon_1 \sim \mathcal{N}(0, 0.03^2)$ and $\epsilon_2 \sim \mathcal{N}(0, 0.01^2)$.

We sample a batch of data points in each training iteration and randomly partition context points and target points for learning. In the testing phase, we draw 100 data points from this mixture of distributions with 15 random data points selected as the context. The fitting results for one sampled mixture curve are shown in Fig. (8).

It can be seen that both CNPs and FCRL display similar patterns, overestimating the uncertainty in the mixture curve of the second component. NPs show intermediate performance and still fail to match context points well. As for MoE-NPs, with help of predicted assignment variables parameters $e_* = \texttt{one\_hot}[\arg_{k\in\{1,2\}} \max \alpha_k]$, we set the number of experts as two and partition data points to visualize predictive distributions $p_\theta(y_*|x_*, z_*)$. The MoE-NP can precisely separate mixture components inside the dataset and provides more reliable uncertainty.

Furthermore, we can understand the assignment latent variables from an entropy perspective. Remember that the role of the discrete latent variable e is to assign the diverse functional prior $z_{1:K}$ to a given data point. With the learned conditional prior $p_{\phi_{2,2}}(e|z_{1:K}, x_i)$ for a data point x_i, we can quantify the uncertainty of assignment via the entropy of such a Bernoulli latent variable $\mathbb{H}[e]$.

$$\mathbb{H}[e] = \sum_{k=1}^{K} -p_{\phi_{2,2}}(e_k = 1|z_{1:K}, x_i) \ln p_{\phi_{2,2}}(e_k = 1|z_{1:K}, x_i) = \sum_{k=1}^{K} -\alpha_k \ln \alpha_k \qquad (4.15)$$

This has a practical significance in discontinuous functions. For example, in regions close to demarcation points, it should be difficult to judge the best expert $z_{1:K}$ to handle these data points, which means the set of $\mathbb{H}[e]$ theoretically exhibits higher uncertainty. Similarly, in regions without context points, it is hard to determine the function as well.

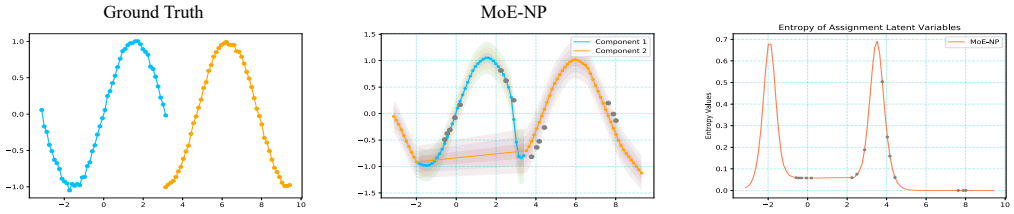


Figure 9: Entropy of Assignment Latent Variables in MoE-NPs. From left to right are respectively the sampled ground truth function, MoE-NP fitting results and the entropy value of discrete latent variable for each data point $\mathbb{H}[p_{\phi_{2,2}}(e|z_{1:K}, x_i)]$.

Interestingly, we observe that MoE-NPs are able to exhibit the above effect on the right side of Fig. (9). The sampled function consists of two components respectively in the interval $[-\pi, \pi]$ and $[\pi, 3\pi]$. The entropy values of our interest are computed via Eq. (4.15). Here $K = 2$ and the learned conditional prior $p_{\phi_{2,2}}(e|z_{1:K}, x_i)$ has highest entropy around the demarcation data point π and the data point -2.0 with no context points nearby. This finding further verifies the role of the assignment latent variable in MoE-NPs.

4.5.3 *Few-Shot Supervised Learning*

We evaluate the performance of models on a system identification task in Acrobot (Killian et al., 2017) and an image completion task in CIFAR10 (Krizhevsky et al., 2009). Both tasks are common benchmarks in the meta learning or NPs literature (Killian et al., 2017; Galashov et al., 2019; Garnelo et al., 2018b,a; Kim et al., 2019a).

Table 7: System Identification Performance in Meta Testing Acrobot Tasks. Shown are mean square errors and standard deviations in fitting meta-testing tasks. Figures in the Table are scaled by multiplying `E-3` for means and standard deviations. The best results are in bold.

CNP	NP	FCRL	MAML	CAVIA	MoE-NP
2.3±0.13	7.2±0.5	2.0±0.15	2.5±0.35	2.0±0.23	**1.4±0.06**

System Identification

For Acrobot systems, different tasks are generated by varying masses of two pendulums. A dataset of state transitions is collected by using a complete random policy to interact with sampled environments. The state consists of continuous as angles and angular velocities $[\theta_1, \theta_1', \theta_2, \theta_2']$. The objective is to predict the resulting state after a selected Torque action from $\{-1, 0, +1\}$. For more details about meta training dataset and environment information, refer to Appendix (B.7.2).

In the meta testing phase, 15 episodes with the length of horizon 200 are collected for each task, and we report the average predictive errors and standard deviations for all transitions. Here we use 50 transitions as the context points to identify the task. As exhibited in Table (7), gradient-based methods, *e.g.* CAVIA and MAML, beat NP in terms of predictive accuracy but show higher variances than all other models. With three experts in modeling, MoE-NPs significantly outperform other baselines in terms of dynamics prediction. Our finding is consistent with observations in (Kégl et al., 2021), where multi-modal distributions are necessary for Acrobot systems. We also illustrate the asymptotic performance of MoE-NPs with increasing the context points in the following Section (4.5.5) Ablation part.

Image Completion

We use the CIFAR10 dataset (Krizhevsky et al., 2009) in this experiment, which is formulated with 32x32 RGB images. In the meta training process, a random number of pixels are masked to complete in images. That is, given the context pixel locations and values $[x_C, y_C]$, we need to learn a map from each 2-D pixel location $x \in [0, 1]^2$ to pixel values $y \in \mathbb{R}^3$. Here two expert latent variables are used in MoE-NPs.

In Fig. (10), we evaluate image completion performance on the test dataset, and the number of context pixels is varied in three levels. It can be found that CAVIA works best in cases with 10 random context pixels or less than 500 ordered context pixels. In other

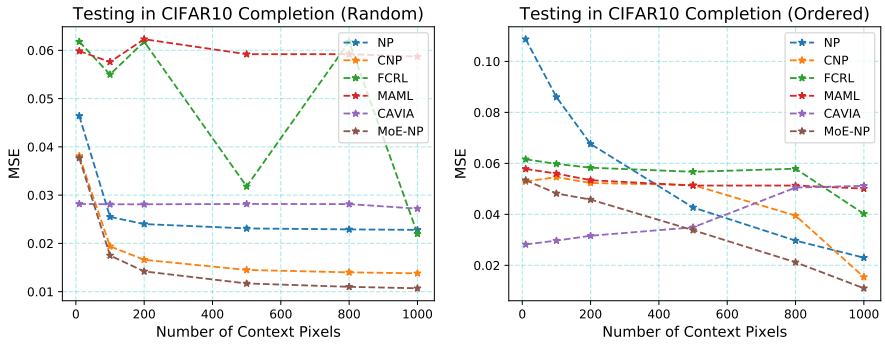


Figure 10: CIFAR10 Completion Performance with Various Number of Context Pixels. The numbers of context points used in prediction are 10, 100, 200, 500, 800, 1000. The left figure is with random context pixels while the right one is with the ordered context pixels.

cases, MoE-NP surpasses all baselines. With more observed pixels, the predictive errors of MoE-NPs can be decreased in both random and ordered context cases. An example of image completion results is displayed in Fig. (11).

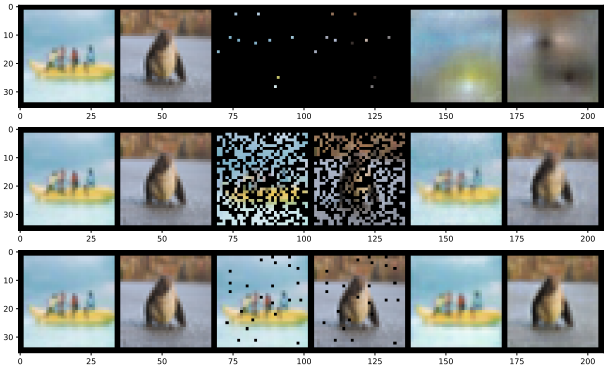


Figure 11: Image Completion Visualization using MoE-NPs. From top to bottom: the number of random context pixels is 10, 500, and 1000. From left to right (every two) are original, masked, and reconstructed images.

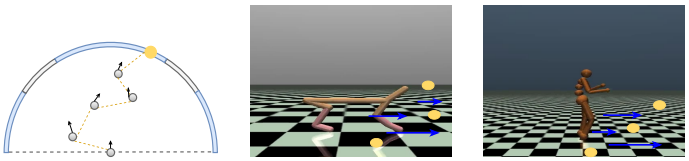
For gradient-based methods, CAVIA and MAML are sensitive to the number of context points and do not exhibit asymptotic performance like that in MoE-NP. NP still suffers underfitting in performance.

4.5.4 *Meta Reinforcement Learning*

To evaluate the meta RL implementation of our model, we conduct the experiments in a 2-D point robot and Mujoco environments (Todorov et al., 2012). Fig. (12) exhibits the environments used in this paper, and we leave more details in Appendix (B.7.1).

2D Navigation Results

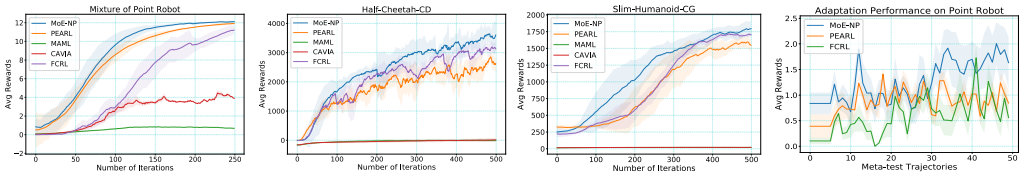
For 2-D point robot tasks, the agent needs to navigate with sparse rewards. The navigation goal of a task is sampled from a mixture of arcs in a semi-circle in Fig. (12.a) during meta training processes.



(a) Point Robot (b) H-Cheetah-CD (c) S-Humanoid-CG

Figure 12: Environments for Meta Reinforcement Learning. **In (a)**: Blue arcs are distributions of goals in orange for the robot to reach with sparse rewards. **In (b)/(c)**: Goals in orange and directions in blue are varied in tasks.

From Fig. (13.a-c), we can observe the evaluation performance of agents over iterations. For gradient-based methods, CAVIA shows better performance than MAML in exploration, but both are weaker than context-based baselines. MoE-NPs can converge earlier with fewer training samples and show a slight advantage over vanilla PEARL. In particular, we test the asymptotic performance in out of distributions (O.O.D.) tasks and show results in Fig. (13.d). We notice O.O.D. tasks are challenging for all algorithms to generalize, but average returns are gradually increased with more trials. PEARL and FCRL achieve comparable rewards, while MoE-NP behaves better in this case.



(a) (b) (c) (d)

Figure 13: Results in Meta Learning Continuous Control. **In (a)/(b)/(c):** Learning curves show tested average returns with variances in 4 runs. 100 transitions are randomly collected from a task-specific memory buffer for point robot environments to infer the posterior. For Mujoco environments, 400 transitions are randomly collected from a task-specific memory buffer to infer the posterior. **In (d):** Fast Adaptation Performance in Meta Testing Point Robot Environments. The collected episodes are gradually increased to 50, and the average returns together with variances are visualized. 5 goals are sampled from the white part of arcs in Fig. (12.a).

Locomotion Results

Half Cheetah-Complex-Direction (H-Cheetah-CD) and Slim Humanoid-Complex-Goal (S-Humanoid-CG) tasks correspond to locomotion in complicated environments. Note that multiple directions and goals are involved in tasks.

Fig. (13) illustrates the performance of learned policies in meta learning tasks. In H-Cheetah-CD, MoE-NP shows a slight advantage over FCRL, and it exhibits comparable performance in S-Humanoid-CG. In both environments, MoE-NP and FCRL outperform other baselines. This implies the importance of functional representations for task-specific value functions. Either contrastive or multiple functional priors lead to better exploration and have the potential to boost performance in continuous control. For

gradient-based methods, observations show that they can easily get stuck in the local optimal (Rakelly et al., 2019; Li et al., 2019).

4.5.5 *Ablation Studies*

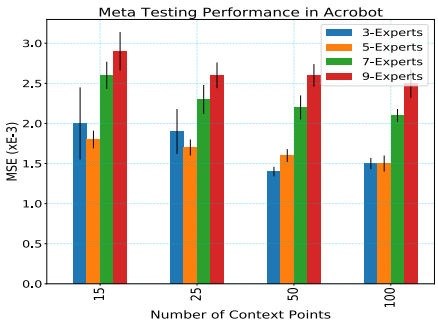


Figure 14: Predictive Performance of MoE-NPs in Acrobot Meta Testing Processes using Varying Numbers of Expert Latent Variables and Context Points. The scale for mean square errors together with standard deviations is E-3.

Number of Experts

We examine the influence of the number of experts in meta-trained MoE-NPs, and system identification in the Acrobot system is selected as an example here. As displayed in Fig. (14), the number of experts are 3, 5, 7 and 9 in different MoE-NPs. Here we test the predictive performance of meta-trained MoE-NP by varying the number of transitions. We set respectively 15, 25, 50, and 100 transition samples as the number of context points to identify the system. All settings for meta testing processes are already described in Section (4.5.3). It can be seen that when the expert number is 5, the predictive performance is largely enhanced with the increase of context points and the variance shrinks accordingly. But with more experts, *e.g.* greater than 5, MoE-NPs exhibit higher predictive errors and show no significant performance improvement with the increase of the number of context points. These suggest that increasing the number of experts beyond a certain point can deteriorate the predictive performance of the MoE-NPs.

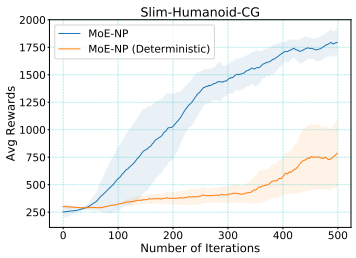


Figure 15: Ablation Performance in S-Humanoid-CG. Learning curves display tested average returns with variances in 4 runs.

Latent Variables in Meta RL

As mentioned in Preliminaries Section, using a latent variable can induce task-specific optimal value functions. Here we take the S-Humanoid-CG as the example, and the expert encoder of MoE-NPs (Deterministic) is set to the deterministic. In Fig. (15), we observe the performance degrades greatly using deterministic expert latent variables. These further verify findings in PEARL (Rakelly et al., 2019). The randomness of value function distributions captures task uncertainty and encourages more efficient exploration.

4.6 CONCLUSION & DISCUSSION

Technical Discussions. In this chapter, we have developed a new variant of NP models by introducing multiple expert latent variables. Our work illustrates the roles of different latent variables in MoE-NPs for meta learning tasks. MoE-NPs are able to separate data points from different clusters of stochastic processes and exhibit superior performance in few-shot supervised learning tasks. Also, MoE-NPs are consistently among the best methods in meta learning continuous control.

Existing Limitations. Though the developed model provides more expressive functional priors, the appropriate number of experts is still hard to determine. Also, the mechanism of gradually incorporating new expert latent variables has not been explored and this raises concerns in additional computational cost and more effective inference.

Future Extensions. Here we provide a couple of heuristics to determine optimal number of experts for MoE-NPs in the future. Information metrics, *e.g.* Bayesian information criterion, can be incorporated in modeling. Another way is to place priors over distributions of discrete latent variables like that in hierarchical Dirichlet processes (Teh et al., 2006) and select the optimal number of experts in a Bayesian way.

5

GRAPH STRUCTURED FUNCTIONAL REPRESENTATION FOR DATA EFFICIENT CONTROL

In the previous chapters (3)/(4), we explored the use of hierarchical and mixture inductive biases in neural processes, which are appropriate for different functional representation problems. This chapter considers the representation of value or policy functions in data efficient sequential decision-making tasks. To achieve this, we employ graph structured neural networks to formulate dynamics models and policies. Our method is able to learn representations of optimal policy functions with high cumulative rewards.

5.1 INTRODUCTION

Reinforcement learning (RL) has been successfully applied to several complicated tasks and achieved remarkable performance, even surpassing outstanding human players in a variety of domains (Mnih et al., 2015; Silver et al., 2017; Vinyals et al., 2019). Exploiting and exploitation can theoretically address a collection of sequential decision-making problems in this paradigm.

Research Motivations. As a cutting-edge research topic, there remain long-standing challenges when putting RL into practice. In particular, these can be viewed from three aspects: i) *data efficiency*, the prevalent branch of RL algorithms as model-free reinforcement learning (MFRL) poses great demands on massive interactions with an environment, making it unrealistic to conduct in most real-world applications (Sutton and Barto, 2018; Chua et al., 2018). ii) *robustness to unseen environments*, when an environment of interest drifts in terms of dynamics or reward mechanisms, previously learned skills suffer the risk of poor generalization (Jing et al., 2018; Clavera et al., 2019). Furthermore, dynamics mismatch easily leads to Sim2Real problems (Peng et al., 2018). iii) *instantaneously planning*, either model predictive control or calibration in previously learned policies consumes additional time in execution phases (Wang and Ba, 2019). Moreover, critical issues might arise in real-time decision-making missions with strict time constraints, *e.g.*, autonomous driving.

Developed Methods. To address the above mentioned concerns, we propose a graph structured surrogate model (GSSM) with an amortized policy search strategy. The work is within the framework of Model-based Meta Reinforcement Learning (MBMRL) (Nagabandi et al., 2019; Sæmundsson et al., 2018; Killian et al., 2017; Lee et al., 2020b). Our approach attempts to improve dynamics prediction, accelerate policy learning and enable fast adaptation across tasks via latent variables. Importantly, unlike most existing

MBMRL methods using time-expensive derivative-free algorithms in model predictive control, we look for the representation of optimal policies and our developed amortized policies do not require adaptation time when faced with a new task.

Our primary contributions are two-fold, respectively in designing flexible meta dynamics models and attaining fast adaptation in policy search in MBMRL:

- **Graph Structured Surrogate Models.** We develop a novel graph structured dynamics model across tasks, which enables the effective encoding of memories and abstracting environments in a latent space. Compared to related work (Kim et al., 2019a), ours is more lightweight but achieves comparable or better performance in forecasting dynamics.
- **Amortized Meta Model-based Policy Search.** We propose a new meta model-based policy search strategy that learns latent variable conditioned policies. This enables fast adaptation without additional policy gradient updates and significantly improves policy performance. Interpretations are given from a perspective of posterior sampling (Osband et al., 2013).

5.2 RELATED WORK

As already mentioned, several critical bottlenecks restrict universal applications of RL algorithms. In terms of mastering new skills rapidly, meta learning is an ideal paradigm to achieve with a few instances. As for data efficiency, both model-based reinforcement learning (MBRL) and meta learning can reduce sample complexity.

Meta Learning. The core of meta learning is discovering common implicit structures across a collection of similar tasks and then generalizing such knowledge to new scenarios. Leveraging knowledge from a meta learner to a task-specific learner is called *fast adaptation*. Two strategies are commonly used for meta learning: Gradient-based Meta-Learning and Contextual Meta-Learning. A representative framework for gradient-based meta-learning is model agnostic meta learning (MAML) (Finn et al., 2017; Flennerhag et al., 2019; Lee and Choi, 2018), where both a meta learner and an adaptor are derived via gradient information after a few shots in a specific task. The contextual meta learning algorithms rely on task specific latent variables to identify a task after a few observations. This strategy theoretically does not require gradient adaptation in new tasks but constructing task relevant latent variables is decisive (Garnelo et al., 2018a,b; Hausman et al., 2018).

Model-based Reinforcement Learning. The key to applications within an RL framework is how to boost sample efficiency, and MBRL serves the role of approximating a target environment for the agent to interact with. In an environment with unknown dynamics, MBRL either learns a deterministic map or a distribution of transitions $p(\Delta s|[s,a])$. Generally, deterministic modeling on dynamical systems does not involve random variables in the hidden units, and some auto-regressive neural network structures are typical in this family (Leibfried et al., 2016; Nagabandi et al., 2017; Amos et al., 2018; van der Pol et al., 2020). Stochastic modeling on dynamical systems is mainly formulated by

incorporating uncertainty in system parameters and observation noise (Deisenroth and Rasmussen, 2011; Kamthe and Deisenroth, 2017; Hafner et al., 2018; Chua et al., 2018).

Meta Reinforcement Learning. Most of meta RL algorithms follow a model-free paradigm, e.g. MAESN (Gupta et al., 2018), RL2 (Duan et al., 2016), Learn2Learn (Wang et al., 2016) and PEARL (Rakelly et al., 2019). However, experimental results show that these methods work poorly with limited training samples. To obtain satisfying performance with lower sample complexity, researchers focus on combining meta learning and MBRL. Nagabandi et al. (2019) takes a gradient-based strategy as MAML and alleviates the gap of Sim2Real. In (Sæmundsson et al., 2018), Gaussian process latent variable models perform task inference and learn dynamics across tasks. CaDM (Lee et al., 2020b) is a novel SOTA MBMRL method that includes forward and backward models to utilize sequential dynamics more effectively. Another model strongly related to ours is in (Galashov et al., 2019), where neural processes (NPs) are used to identify dynamics of tasks, but it requires to re-train or fine-tune parameterized policies via gradient updates in new tasks. Note that most of these MBMRL methods focus on *fast adaptation* in *dynamics models*. These either make use of *derivative-free* algorithms for model predictive control or re-train policies in separate tasks, which requires an additional computational cost on environments with higher dimensionality (Wang and Ba, 2019). In AdMRL (Lin et al., 2020), task-specific policies are optimized using an implicit function theorem, but it considers the case when dynamics are shared across tasks with different goals. In MIER (Mendonca et al., 2020), labels are updated by querying a neural network of a dynamics system, which is efficient in practice. In our method, we amortize this step and further reduce computations in adaptation.

5.3 PRELIMINARIES

RL's decision-making process is usually characterized by a discrete-time Markov Decision Process (MDP), denoted by $\mathcal{M}$. Given states $s_t \in \mathcal{S}$, actions $a_t \in \mathcal{A}$, policy functions π, state transition distributions $\mathcal{P}$, reward functions $\mathcal{R}$ and a discount factor γ for a step-wise reward, a MDP can be formalized with a tuple of these elements $\mathcal{M}_k = (\mathcal{S}, \mathcal{A}, \mathcal{P}_k, \mathcal{R}_k, \gamma)$. The return of cumulative rewards is a summation of discounted reward feedback $r(s_t, a_t)$ along the trajectories $\tau := (s_0, a_0, r_0, \ldots, s_{H-1}, a_{H-1}, r_{H-1}, s_H)$.

The optimization objective in model-free RL methods is to find policies that maximize the expected cumulative rewards over trajectories. In contrast, MBMRL considers a distribution over MDPs $p(\mathcal{M})$, and the goal is to simultaneously build dynamics models and act optimally w.r.t. the learned dynamics models.

5.3.1 *Optimization Objective in MBMRL*

More formally, we study MBMRL problems from the optimization perspective and formulate the following two correlated objectives.

$$\max_{\theta} \mathbb{E}_{\substack{\mathcal{M}\sim p(M) \\ ([s,a],s')\sim\mathcal{M}}} \ln\Big[p_{\theta_{\mathcal{M}}}(s'|[s,a])\Big], \text{ s.t. } p_{\theta_{\mathcal{M}}} = u(\theta, \mathcal{D}^{\text{tr}}_{\mathcal{M}}) \tag{5.1a}$$

$$\max_{\varphi_{\mathcal{M}}} \mathbb{E}_{\substack{s'\sim p_{\theta}{}_{\mathcal{M}}(s'|[s,a]) \\ a\sim\pi_{\varphi_{\mathcal{M}}}}} \left[\sum_{t=0}^{H-1} \gamma^t r_{\mathcal{M}}(s_t, a_t)\right], \ \mathcal{M} \sim p(M) \tag{5.1b}$$

Here Eq. (5.1a) is to maximize the log-likelihood of state-transitions $p(s'|[s,a])$ in a collection of MDPs and u represents a fast adaptation mechanism in $\mathcal{M}$ to learn an updated dynamics model $p_{\theta_{\mathcal{M}}}$ with meta-learned parameters θ and a few transition instances $\mathcal{D}^{\text{tr}}_{\mathcal{M}}$. Eq. (5.1b) corresponds to learning a policy $\pi_{\varphi_{\mathcal{M}}}$ or finding a planning strategy in separate dynamics models. It is worth noting that our objective of MBMRL differs from previous work, and it consists of two phases as *dynamics model learning* and *policy optimization*.

5.3.2 *MBMRL with Latent Variables*

Latent variables play diverse roles in meta RL (Gal et al., 2016; Rakelly et al., 2019). Here we focus on a branch of MBMRL methods, which uses latent variables to help formulate meta dynamics models. A latent variable z is mainly inferred from a few shots of transitions to summarize statistics of a specific environment $\mathcal{M}$ (Garnelo et al., 2018b,a). Then the learned latent variables participate in dynamics prediction $p_\theta(\Delta s|[s,a],z)$ and approximate dynamics of different MDPs (Galashov et al., 2019; Lee et al., 2020b).

The meta learning surrogate model (MLSM) (Galashov et al., 2019) is an example of latent variable MBMRL methods, which is closest to ours in literature. The neural processes (Garnelo et al., 2018b) work as meta dynamics models in MLSM. But we notice that in MLSM: (1) a simple *mean pooling* over context points to obtain latent variables is challenging to utilize the relevance between the context and the target transition samples for all data points' prediction; (2) computationally expensive policy gradient updates are required in policy search when faced with a new task.

5.3.3 *Influence of Model Discrepancy*

For more insights, we define the discrepancy between a distribution over MDPs and a distribution over learned dynamics models using the expected form of the total variance distance as $\mathbb{E}_{\substack{M\sim p(M) \\ (s,a)\sim\nu(s,a)}} \Big[D_{\text{TV}}[P_{\hat{M}}(\cdot|s,a)), P_M(\cdot|s,a)]\Big]$, where $P_{\hat{M}}(\cdot|s,a)$ and $P_M(\cdot|s,a)$ are respectively a learned state transition distribution and a transition distribution in a real environment and $\nu(s,a)$ is a distribution over state action pairs. By extending findings in (Rajeswaran et al., 2020) to meta learning, we can depict a performance gap between dynamics models and real environments under an arbitrary policy in **Lemma** (1).

Lemma 1 *Assume the discrepancy between transition distributions over MDPs and the learned approximated models* $\mathbb{E}_{\substack{M\sim p(M) \\ (s,a)\sim\nu(s,a)}} \Big[D_{TV}[\hat{P}_M(\cdot|s,a), P_M(\cdot|s,a)]\Big] \leq \epsilon$*, we can estimate the performance gap under a policy* π *as follows,*

$$\mathbb{E}_{M\sim p(M)}\left[|\mathcal{J}_{\hat{M}}(\pi)-\mathcal{J}_{M}(\pi)|\right]\leq\frac{2\epsilon\mathcal{R}_{\max}}{(1-\gamma)^{2}} \tag{5.2}$$

where $\mathcal{R}_{\max}$ is the supremum value of one step reward with discount factor γ, $\hat{M}$ and M are a sampled approximated MDP and a corresponding real MDP, and the expected rewards are $\mathcal{J}_{M}(\pi)=\mathbb{E}_{\substack{s'\sim p_{\mathcal{M}}(s'|[s,a])\\ a\sim\pi(\cdot|s)}}\Big[\sum_{t=0}^{\infty}\gamma^{t}r_{\mathcal{M}(s_t,a_t)}\Big]$.

Theorem 2 *Suppose a bounded dynamics models' discrepancy as*

$$\mathbb{E}_{\substack{M\sim p(M)\\ (s,a)\sim\nu(s,a)}}\Big[D_{TV}[\hat{P}_{M}(\cdot|s,a),P_{M}(\cdot|s,a)]\Big]\leq\epsilon$$

over a MDP distribution and optimal policies for any MDP $\mathcal{M}$ and its approximation $\hat{\mathcal{M}}$ are respectively $\pi_{\hat{M}}$ and π_{M}. Then the regret bound is estimated as follows.

$$\mathbb{E}_{M\sim p(M)}\left[\mathcal{J}_{M}(\pi_{\hat{M}})\right]\geq\mathbb{E}_{M\sim p(M)}\left[\mathcal{J}_{M}(\pi_{M})\right]-\frac{4\epsilon\mathcal{R}_{\max}}{(1-\gamma)^{2}} \tag{5.3}$$

The proposition in **Lemma** (1) is to disentangle the relationship between dynamics learning and policy performance with a performance difference, suitable for most MBMRL algorithms. In addition, the inherent model bias in MBRL tends to bring catastrophic failure, and in meta learning scenarios **Theorem** (2) implies that potential expected regret can be well bounded or minimized if the discrepancy in a dynamics model sense is small enough. The proof is given in Appendix (C.2/C.3).

5.4 METHODS

In this section, we aim to address the shortcomings of insufficiently expressive models and expensive policy gradient optimization by MLSM.

In order to do so, we first develop a graph structured surrogate model to learn representations of latent variables via message passing, which better approximates the local dynamics of MDPs. Then an amortized policy search strategy is introduced to enable fast policy adaptation without gradient updates in new tasks.

In GSSM, two types of latent variables are learned. Fig. (16), which will be explained fully in the following section, illustrates target latent variables z_t to predict individual state transition and a global latent variable z_c to encode the context for policies to condition. For the sake of simplicity, we denote the dynamics model input by $x=[s,a]$ and the dynamics model output by $y=\Delta s$.

5.4.1 *Graph Structured Latent Variables*

For the transition dataset from a task, we have a set of context points $[x_c,y_c]$ and the target point $[x_t,y_t]$. Similar to other context-based meta learning algorithms (Garnelo et al., 2018b), $[x_c,y_c]$ are observed state transitions used to infer the task. We treat these context points in a form of graph structured dataset $\mathcal{G}=<\mathcal{V},\mathcal{E}>$, which comprises of

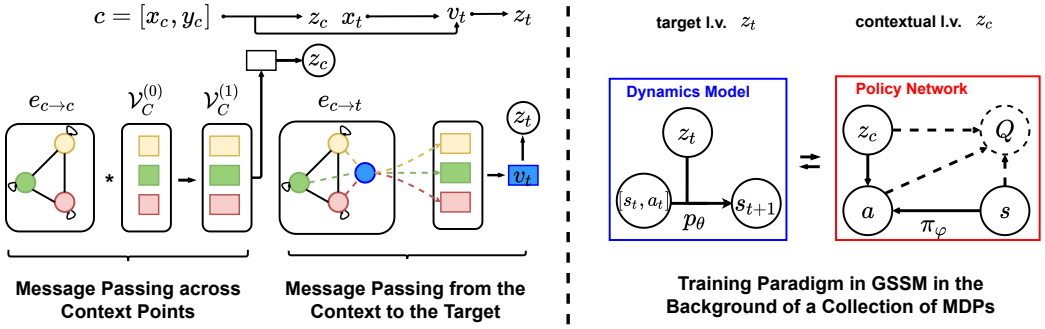


Figure 16: Graph Structured Surrogate Models. On the **Left**: The colored □ denotes a node feature vector in our fully connected graphs with nodes ◯, and lines between ◯ record pairwise similarities $\texttt{sim}$. $\mathcal{V}_C^{(0)}$ denotes the initial feature matrix of all context nodes. Information Flows in **Dynamics Models** from the left to the right describe the Message Passing between context points c (Edges in solid lines $e_{c\to c}$) and to the **target point** x_t (Edges in dashed lines $e_{c\to t}$) to obtain the transformed node feature matrix $\mathcal{V}_C^{(1)}$. On the **Right**: In amortized policy search, latent variables participate in both approximate **Dynamics Models** and **Policy Networks** (Dashed elements are involved in the module when using Actor-Critic frameworks and Double arrows mean interactions to learn amortized policies).

a collection of vertices $\mathcal{V} = [x_c, y_c]$ and relational edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ (notations edge e and vertex value v in Fig. (16)).

Here a fully connected graph is built to characterize the pairwise relationship. The value as the initial node feature vector refers to $[x_i, y_i]$, and the construction of a graph Laplacian matrix is based on the pairwise similarities between samples $\texttt{sim}(x_i, x_j) = \frac{\langle u(x_i), u(x_j)\rangle}{\|u(x_i)\|_2 \cdot \|u(x_j)\|_2}$, where $u(x)$ is the embedding of a sample input x using neural networks, and the notation $\langle \cdot, \cdot \rangle$ means a dot product. All of these are illustrated on the left side of Fig. (16).

Message Passing between Context Points

This process is to deliver and aggregate neighborhood information for node representations, which can be specified via the normalized Laplacian as follows.

$$\hat{s}_{ij} = \frac{\exp\Big\{\big(\beta \cdot \texttt{sim}(x_i, x_j)\big)\Big\}}{\sum_{j \in \mathcal{O}_i} \exp\Big\{\big(\beta \cdot \texttt{sim}(x_i, x_j)\big)\Big\}} \tag{5.4a}$$

$$h_i^{(l+1)} = \sigma\left(W^{(l)} h_i^{(l)} + \sum_{j \in \mathcal{O}_i} \hat{s}_{ij} W^{(l)} h_j^{(l)}\right) \tag{5.4b}$$

where β is a tunable parameter for pairwise similarities, $W^{(l)}$ is the network parameter for node feature transformations in l-th layer, $\hat{s}_{ij}$ is the element in a normalized Laplacian

matrix L, and $h_i^{(l)}$ is the l-th intermediate node feature vector after message passing from sample i's neighborhoods O_i to itself.

In Eq. (5.4), a self-loop is added for message passing. The node j's feature vector after the final message passing is denoted by h_j. With the operation of message passing in Eq. (5.5), a learned representation for each node summarizes statistics of interactions,

$$v_i = \sum_{j \in O_i} \hat{s}_{ij} h_j, \; g_c = \bigoplus_{i \in \mathcal{V}} v_i, \; q_{\phi_2}(z_c) = \mathcal{N}\Big(\mu(g_c), \Sigma(g_c)\Big) \tag{5.5}$$

where $\bigoplus$ denotes a mean pooling operation over all node feature vectors, and the last term describes the amortized distribution (Zhang et al., 2018) for context points.

Message Passing from the Context to the Target

This process is to transmit task-beneficial information from the context to the target. And the representation of the node i is v_i in Eq. (5.5). Considering relevance to the target point $[x_t, y_t]$ varies from instance to instance, we compute the weight for each context point x_i, denoted by $\hat{s}_{it}$, via Eq. (5.4.a) to measure the relevance. Then the aggregated context message can be represented as g_t in a weighted way.

$$g_t = \sum_{i \in O_t} \hat{s}_{it} v_i, \quad q_{\phi_1}(z_t) = \mathcal{N}\Big(\mu(g_t), \Sigma(g_t)\Big) \tag{5.6}$$

After the message passing from the context to the target, g_t is further mapped into mean and variance parameters of a proposal distribution $q_{\phi_1}(z_t)$ using neural networks, as displayed on the right side of Eq. (5.6). Here we employ mean field amortized inference, using diagonal Gaussian distributions for the convenience of computations. Also, note that ϕ_1 and ϕ_2 share part of graph neural net parameters, *e.g.* parameters in feature embeddings, but denote separate variational parameters.

5.4.2 *Approximate Inference & Scalable Training in GSSM*

To learn system dynamics with latent variables, we need to specify an objective in optimization together with a predictive distribution $p(y_t|x_t, x_c, y_c)$. Here we sample the context data points $[x_c, y_c]$ together with the target $[x_t, y_t]$ from meta learning dataset $p(D)$.

Though the exact inference for this predictive distribution is intractable, a plausible way is to use the above mentioned variational distribution $q_{\phi_1}(z_t|x_t, x_c, y_c)$ in Eq. (5.6). As a result, the evidence lower bound (ELBO) is formulated on the right side of Eq. (5.7).

$$\begin{aligned} \mathbb{E}_{p(D)}\Big[\ln p(y_t|x_t, x_c, y_c)\Big] \geq \mathbb{E}_{p(D)}\Big[\mathbb{E}_{q_{\phi_1}}[\ln p_\theta(y_t|x_t, z_t)]\Big] \\ -\mathbb{E}_{p(D)}\left[D_{KL}[q_{\phi_1}(z_t|x_t, x_c, y_c) \parallel p(z_t)]\right] \end{aligned} \tag{5.7}$$

Similar to (Denton and Fergus, 2018; Pertsch et al., 2020; Garnelo et al., 2018b), a variational distribution $q_{\phi_2}(z_c|x_c, y_c)$ is selected as a prior distribution $p(z_t)$ in ELBO to specify a dynamical system from context points and further *used in the following amortized policy search*. As a result, the induced objective with a learned prior distribution is as follows.

$$\begin{aligned}\mathbb{E}_{p(D)}\Big[\ln p(y_t|x_t, x_c, y_c)\Big] \geq \mathbb{E}_{p(D)}\Big[&\mathbb{E}_{q_{\phi_1}}[\ln p_\theta(y_t|x_t, z_t)] \\ &-D_{KL}[q_{\phi_1}(z_t|x_t, x_c, y_c) \parallel q_{\phi_2}(z_c|x_c, y_c)]\Big]\end{aligned} \tag{5.8}$$

In implementations, the Monte Carlo estimation is performed for the negative ELBO to obtain Eq. (5.9),

$$\begin{aligned}\mathcal{L}(\theta, \phi_1, \phi_2) = -\frac{1}{K}\sum_{b=1}^{B}\sum_{k=1}^{K}\ln p_\theta(y_t^{(b)}|x_t^{(b)}, z_t^{(b,k)}) \\ +D_{KL}\Big[q_{\phi_1}(z_t^{(b)}|x_t^{(b)}, x_c^{(b)}, y_c^{(b)}) \parallel q_{\phi_2}(z_c^{(b)}|x_c^{(b)}, y_c^{(b)})\Big]\end{aligned} \tag{5.9}$$

where B is the batch size of samples in meta training, K is the number of particles in estimation, and latent variable values are sampled from the approximate posterior $z_t^{(b,k)} \sim q_{\phi_1}(z_t^{(b)}|x_t^{(b)}, x_c^{(b)}, y_c^{(b)})$.

Similarly, when it comes to prediction using the learned dynamics model, the Monte Carlo estimator can be applied again to derive a predictive distribution in Eq. (5.10) with the approximate posterior q_{ϕ_1} and collected context points $[x_c, y_c]$.

$$p(y_t|x_t, x_c, y_c) = \int q_{\phi_1}(z_t|x_t, x_c, y_c)p_\theta(y_t|x_t, z_t)dz_t \approx \frac{1}{K}\sum_{k=1}^{K} p_\theta(y_t^{(k)}|x_t, z_t^{(k)}) \tag{5.10}$$

5.4.3 *Amortized Policy Search*

Once a dynamics model is learned, policy search can be executed by interacting with a learned dynamics model. Here we concentrate on *fast adaptation of policies* in MBMRL and introduce the concept of *amortized policy search*. This differs from previous planning or policy optimization objectives in separate dynamics models.

To this end, we utilize the strategy of posterior sampling (Osband et al., 2013; Rakelly et al., 2019) to capture task-specific policies. Specifically, a collection of approximate MDPs are sampled from the posterior of task-specific dynamics models, and the agent acts optimally w.r.t. these models. Finding optimal policies for different tasks are time consuming in previous approaches, which require either *re-training* or *fine-tuning meta-learned policies* $\pi_\varphi(a|s)$ in separate dynamics models (Galashov et al., 2019). So we propose to induce task-specific policies $\pi_\varphi(a|[s, z_c])$ by sampling z_c from task relevant latent variable distributions $q_{\phi_2}(z_c|x_c, y_c)$, and optimize policies w.r.t. sampled approximate MDPs. The process of finding these task-specific optimal policies is termed

as *amortized policy search* and the obtained policy $\pi_\varphi(a|[s, z_c])$ after meta training is called an *amortized policy*.

In our settings, meta model-based policy search is considered in a distribution of learned approximate dynamics models $p(\hat{\mathcal{M}}; \theta, \phi)$. A parameterized policy π_φ is used to collect trajectories τ from a learned dynamics model and evaluate rewards of policies in Eq. (5.11) to maximize.

$$\mathcal{J}(\varphi; \theta, \phi) = \iint p(\hat{\mathcal{M}}; \theta, \phi) p(\tau|\hat{\mathcal{M}}; \varphi, \phi) \mathcal{R}(\tau) d\tau d\hat{\mathcal{M}} \tag{5.11}$$

The following two model-based policy search strategies are commonly used in this domain, so we modify these to enable the use of amortized policies in MBMRL. Importantly, we take more interest in computing gradients of policies w.r.t. policy parameters φ.

- **Direct policy search via BPTT.** The objective of back-propagation through time (BPTT) (Deisenroth and Rasmussen, 2011; Parmas et al., 2018) in MBMRL can be written in the form of Monte Carlo estimation as follows.

$$\nabla_\varphi \mathcal{J}(\varphi) \approx \frac{1}{BK} \sum_{b=1}^{B} \sum_{k=1}^{K} \left(\nabla_\varphi \sum_{t=0}^{T-1} \gamma^t r_{t+1}^{(b,k)} \right) \tag{5.12}$$

  where $\tau = [s_0, a_0, s_1, r_1, \dots]$, $a_t \sim \pi_\varphi(\cdot|[s_t, z_c])$, $s_{t+1} \sim p_\theta(s_{t+1}|[s_t, a_t], z_t)$, B is the number of batch in tasks, and K is the number of sampled simulated trajectories for each task.

- **Actor-Critic policy gradient optimization.** The induced policy gradient w.r.t Eq. (5.11) is estimated as Eq. (5.13) under our amortized policy search method, where A_t is the computed advantage function.

$$\nabla_\varphi \mathcal{J}(\varphi) = \mathbb{E}_{\substack{\hat{\mathcal{M}} \sim p(\hat{\mathcal{M}}; \theta, \phi) \\ \tau \sim p(\tau|\hat{\mathcal{M}}; \varphi, \phi)}} [\sum_{t=0}^{T-1} \nabla_\varphi \ln \pi(a_t|[s_t, z_c]) \cdot A_t([s_t, z_c], a_t)] \tag{5.13}$$

Besides, the optimization of value function approximation is inside policy search during meta-training processes via gradient updates. For the sake of simplicity, we skip this step in Algorithm (6), and the optimization is executed in `step` (10) as default. For more details about Actor-Critic settings, please refer to Appendix (C.7) for more details.

Note that the universal value function approximator augments MDPs with goal information to identify diverse tasks (Schaul et al., 2015). Similarly, in our settings, the task is inferred from context points, and the transition sample is augmented with the inferred task/goal as $\{([s_t, z_c], a_t, r_t)\}_{t=1}^{T}$. The approximate value function $Q_{\pi_\varphi}([s, z_c], a)$ or $V_{\pi_\varphi}([s, z_c])$ is task-specific. In this case, our proposed amortized policy search can be interpreted as finding a universal value function approximator in MDPs of various dynamics.

Algorithm 6: Meta-Training Process.

Input : MDP distribution $\rho(\mathcal{M})$; Batch of tasks $\mathcal{B}$; Exploration policy π_e.
Output : Trained parameters ϕ, θ and φ.

Initialize parameters ϕ, θ and φ;
while *Meta-Training not Completed* **do**
    // meta train dynamics models
    Sample a batch of tasks $\{\mathcal{M}_k\}_{k=1}^{\mathcal{B}}$ from a distribution $\rho(\mathcal{M})$;
    Perform roll-outs to collect transition dataset $\mathcal{D}^{\mathcal{B}} = \{\mathcal{D}_k\}_{k=1}^{\mathcal{B}}$ with π_e;
    Optimize $\{\phi, \theta\}$ on $\mathcal{D}^{\mathcal{B}}$ in Eq. (5.10) to obtain $\{\hat{\mathcal{M}}_k\}_{k=1}^{\mathcal{B}}$;
    // meta model-based policy search
    for $i = 1, 2, \ldots, N$ **do**
        Sample initial states from $\{\hat{\mathcal{M}}_k\}_{k=1}^{\mathcal{B}}$;
        Collect episodes by interacting with $\{\hat{\mathcal{M}}_k\}_{k=1}^{\mathcal{B}}$ using π_φ and update dynamics buffers $\mathcal{D}^{\mathcal{B}}$;
        Evaluate rewards in Eq. (5.11);
        Optimize the policy π_φ: $\varphi \leftarrow \varphi + \alpha \nabla_\varphi \mathcal{J}$.
    end
end

Algorithm 7: Meta-Testing Phases.

Input : Meta-trained ϕ, θ and φ ; Null buffer $\mathcal{B}$; Adaptation steps K.
Output : Cumulative rewards of episodes.

Sample a testing task $\mathcal{M}_* \sim \rho(\mathcal{M})$;
if *Use GSSM and Amortized Policy Search* **then**
    Run π_e to collect the memory $\tau : \mathcal{B} \leftarrow \mathcal{B} \cup \{\tau\}$;
    Evaluate $\pi_\varphi(a|s,z)$ with $z \sim q(z_c|\mathcal{B})$ in Eq. (5.11) in $\mathcal{M}_*$.
else
    Run π_e to collect the memory $\tau : \mathcal{B} \leftarrow \mathcal{B} \cup \{\tau\}$;
    for $i = 1, 2, \ldots, K$ **do**
        Sample an initial state from the learned $\hat{\mathcal{M}}_*$;
        Collect an episode in $\hat{\mathcal{M}}_*$ using $\pi_\varphi(a|s)$;
        Evaluate return in Eq. (5.11);
        Optimize the policy π_φ: $\varphi \leftarrow \varphi + \alpha \nabla_\varphi \mathcal{J}$.
    end
    Evaluate fine-tuned $\pi_\varphi(a|s)$ in $\mathcal{M}_*$.
end

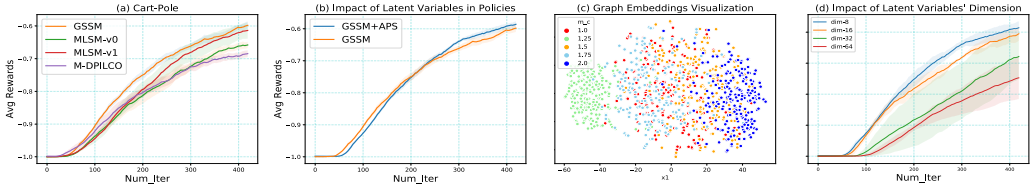


Figure 17: Experimental Results on Cart-Pole Tasks. (a) Learning curves of MBMRL algorithms using non-latent variable conditioned policies. (b) Learning curves of GSSM using (non-)latent variable conditioned policies. (c) t-SNE visualizations of latent variables in GSSM (samples are from 5 tasks with a pole mass as 1.0 and cart masses as $\{1.0, 1.25, 1.5, 1.75, 2.0\}$). (d) Learning curves of GSSM+APS with different dimensions of latent variables. For all learning curves, the averaged rewards are tested after each `iter` in an offline way, and results indicate means and a standard error of the mean in 5 runs.

5.5 EXPERIMENTS

We perform experiments in environments with diverse task-specific dynamics to evaluate our approach. Meanwhile the implementation of our developed approach is available in Appendix (C.8).

5.5.1 *General Settings*

In all MBMRL related experiments, meta training and testing phases respectively follow that in Algorithm (6)/(7), where $\hat{\mathcal{M}}$ is an approximate dynamics model. Similar to that in (Galashov et al., 2019), the exploratory policy π_e is completely random without parameters to initialize the transition buffer or collect transitions for identifying the task in meta testing phases.

Some baseline methods include:

- **L2A** (Nagabandi et al., 2019). As a gradient-based meta RL approach, the Learning to Adapt (L2A) utilizes a MAML paradigm to learn dynamics and adaptation strategies.
- **MLSM-v0** (Galashov et al., 2019). Meta Learning Surrogate Model (MLSM) makes use of neural processes in MBMRL, where latent variables are incorporated to identify tasks.
- **MLSM-v1** (Galashov et al., 2019; Kim et al., 2019a). This is a boosted version of MLSM-v0, where an attention neural network is added to learn sample dependent latent variables.
- **M-DPILCO** (Gal et al., 2016). Deep PILCO, which uses Bayesian neural networks (BNNs) to fit dynamics. Ensemble of episodes from BNNs are used for policy optimization.

For GSSM/M-DPILCO/MLSM-v0/MLSM-v1, we use the same policy search strategy: a policy $\pi_\varphi(a|s)$ is optimized across a collection of approximate dynamics models in

Algorithm (6), and in testing processes, this policy is fine-tuned via policy gradient updates as fast adaptation in separate dynamics models like that in Algorithm (7). For GSSM+APS, we use a latent variable conditioned policy $\pi_\varphi(a|s, z_c)$ to enable amortized policy search (APS) in GSSM meta dynamics models. As for L2A, the implementation follows that in (Nagabandi et al., 2019), each learned dynamics model after fast adaptation via gradient updates is used to plan separately.

With these models, we use formerly introduced meta model-based policy search strategies to combine: (i) direct policy search trained via BPTT (Parmas et al., 2018) (only applied to Cart-Pole environments) (ii) actor-critic policy search using proximal policy optimization (PPO) (Schulman et al., 2017) (applied to all the other environments).

Meanwhile, **DR-PPO** is included as a model-free RL baseline, where PPO is trained across tasks via Domain Randomization. Another algorithm as the probabilistic embedding for actor-critic RL (referred to as **PE-PPO**) is also introduced in comparisons, which follows the same implementation in PEARL algorithms (Rakelly et al., 2019). More details about task settings and available PyTorch modules are given in Appendix (C.8).

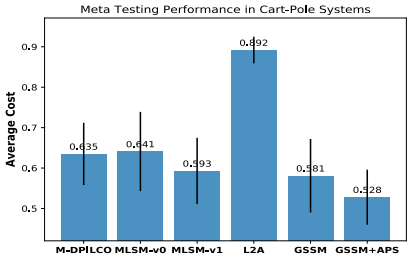


Figure 18: Meta Testing Performance. The average cost is negative of average rewards, and standard deviations are attached as error bars. For methods except GSSM+APS, *additional time* is required to perform gradient-based adaptation in policies.

5.5.2 *Cart-Pole Systems*

At first, we evaluate MBMRL models on a Cart-Pole Swing-Up task. The physics system can be found in (Gal et al., 2016; Galashov et al., 2019), and meta tasks are generated in the following way. We respectively sample masses of a cart m_c and a pole m_p from uniform distributions $\mathcal{U}[1.0, 2.0]$ and $\mathcal{U}[0.7, 1.0]$. The mission is to perform actions to reach the goal with the end of the pole. The state is $[x_c, \theta, x_c', \theta']$, while the action corresponds to the force in a continuous interval $a \sim [-10, +10]$ N. The horizon in episodes is set to be 25, the same as that in former works.

In **meta training** processes, various tasks are sampled during iterations, and results are averaged step-wise rewards. In Fig. (17.a), with the same policy search strategies (non-amortized policies), we observe GSSM shows best performance with lower variances in learning curves, followed by MLSM-v1 with larger variances. Also note that GSSM's model complexity is more lightweight than AttnNP in MLSM-v1 while retaining satisfying dynamics prediction capability (Refer to Table (25)/Table (24)). In Fig. (17.b), when amortized policies are combined with GSSM, the performance is

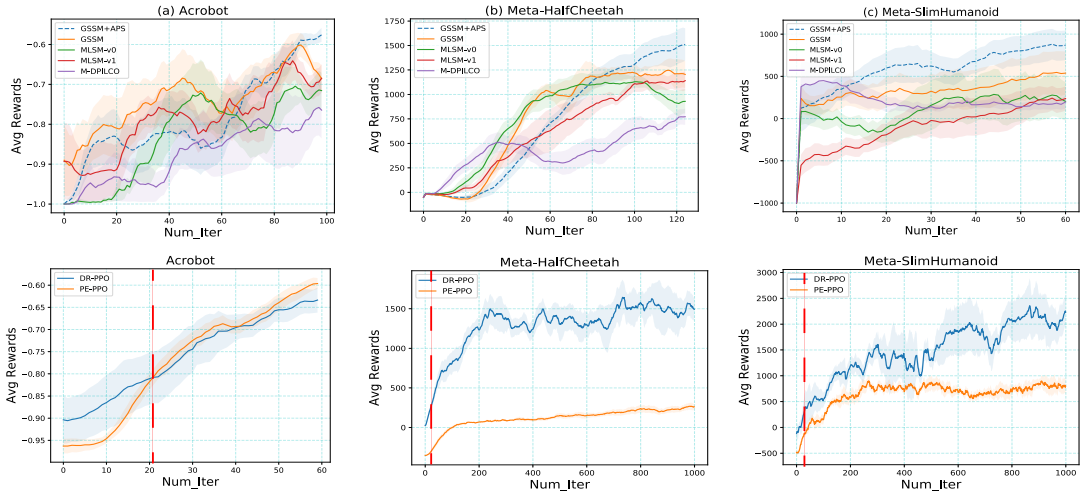


Figure 19: Performance of Policies in Meta-Training Processes. Figures in the first row are MBMRL results, while those in the second row are MFRL results (Red vertical dotted lines indicate the threshold of required time-steps used to train MBMRL baselines.). Here environments are varied in terms of dynamics during iterations. The average rewards are tested after each `iter` in an offline way, and results indicate means and corresponding standard errors of the mean in 5 runs.

further advanced. In Fig. (17.c), we sample latent values by encoding trajectories from 5 tasks to visualize using t-SNE (Van der Maaten and Hinton, 2008). It can be seen that the latent embeddings of different tasks formulate several clusters, which reveal cart masses' impact on latent variables. In Fig. (17.d), we also vary the dimension of latent variables and find that when amortized policies are used, a lower dimension results in more compact task embeddings and obtains better results with lower variance.

Note that the principal goal of MBMRL is to generalize previously learned skills to unseen tasks, so we stress the importance of the performance in **meta testing** processes, which is more appropriate to assess the generalization capability. Here 50 unseen tasks are sampled to validate the performance (each task with 50 episodes) and averaged results are displayed in Fig. (18). We observe that GSSM and MLSM-v1 are comparable

Table 8: Meta-testing Results using Model-free Baselines. MUL records multiple of required samples used in MBMRL.

Env	Mul	DR-PPO	PE-PPO
Acrobot	1x	-0.836±0.047	**-0.828±0.052**
	3x	-0.433±0.043	**-0.420±0.05**
H-Cheetah	1x	**453.4±150**	-44.1±51
	25x	**1360.5±130**	608.2±73
S-Humanoid	1x	**538.5±91**	252.4±260
	25x	**3533.3±110**	1248.1±150

Table 9: Average Rewards in Meta-testing Tasks using Learned Policy Networks. (For each testing task, 50 episodes are sampled and averaged in rewards. Figures behind ± are standard deviations across testing tasks, with bold ones the best.)

Env	GSSM+APS	GSSM	M-DPILCO	MLSM-v0	MLSM-v1	L2A
Acrobot	**-0.478**$_{\pm 0.049}$	-0.506$_{\pm 0.068}$	-0.645$_{\pm 0.06}$	-0.560$_{\pm 0.064}$	-0.524$_{\pm 0.052}$	-0.7775$_{\pm 0.054}$
H-Cheetah	**1597.4**$_{\pm 200}$	1306.6$_{\pm 140}$	862.0$_{\pm 280}$	827.3$_{\pm 190}$	1226.8$_{\pm 64}$	-17.9$_{\pm 130}$
S-Humanoid	**1641.8**$_{\pm 170}$	717.1$_{\pm 130}$	596.0$_{\pm 340}$	285.9$_{\pm 360}$	745.6$_{\pm 150}$	124.9$_{\pm 570}$

in policy performance, but the former has fewer parameters. As for M-DPILCO and MLSM-v0, they show intermediate performance in testing results. With the same type of dynamics models, GSSM+APS does not require adaptation time and reduces **10%** step-wise costs than GSSM. This suggests that amortized policies reveal task relevant information from MDP embeddings and then guide the agent to explore better in separate environments.

5.5.3 *Other Simulation Systems*

Other explorations are performed in more complicated simulation systems. These include Acrobot, Half-Cheetah and Slim-Humanoid. For Acrobot, masses m of two pendulums are respectively drawn from uniform distributions as $\mathcal{U}[0.8, 1.2]$ and $\mathcal{U}[0.8, 1.2]$ to configure different tasks with 200 steps as the default horizon (Killian et al., 2017; Sutton and Barto, 2018). For H(alf)-Cheetah/S(lim)-Humanoid from Mujoco (Todorov et al., 2012), we vary mass scales and damping coefficients for different tasks, where the default horizon is 1000 steps.

Main Results in Meta-training Processes. It can be seen in Fig. (19) that with the equal volume of samples, MBMRL models mostly outperform MFRL baselines. Without the use of amortized policies, GSSM shows an advantage over other MBMRL baselines. For MLSM-v0/MLSMv-1, intermediate rewards are obtained in Acrobot/H-Cheetah. We also observe unstable results using other baselines in S-Humanoid. When amortized policies are employed, GSSM+APS shows significant performance improvement over all baselines, including GSSM. This reflects that task-specific information is beneficial in policy optimization. Especially, GSSM+APS reaches around -0.57, equivalent to model-free performance with 3x fewer time steps in Acrobot. A similar phenomenon can be seen in H-Cheetah, and GSSM+APS mostly starts with poor initialization but gradually surpasses others as latent variables become more and more meaningful.

Main Results in Meta-testing Processes. As exhibited in Table (25), we evaluate predictive performance of dynamics models in unseen tasks. After executing the paired-t test, we find GSSM+APS significantly surpasses MLSM-v0 in Cart-Pole/Acrobot/S-Humanoid and achieves comparable performance to MLSM-v1 in forecasting dynamics. However, GSSM is simpler in terms of model complexities.

The corresponding policy performance is also displayed in Table (9). We also notice that with the same policy search strategy, GSSM outperforms other baselines in Acrobot/H-Cheetah. When amortized policies are used, GSSM+APS is significantly superior to MLSM-v1[1]. These findings reflect strong generalization capability in unseen tasks when combining GSSM and amortized policies. L2A works not so well in our environments even after trying several hyper-parameters, similar to observations in work (Hiraoka et al., 2020; Lee et al., 2020b), and lower rewards could be due to unstable adaptation in dynamics models. Results in model-free cases are also illustrated in Table (8). PE-PPO performs worse than DR-PPO in two environments, even though probabilistic embeddings of tasks join the policy learning. Here we use permutation invariant amortized distributions to formulate $\pi(a|s,z)$ as in PEARL (Rakelly et al., 2019), but PEARL algorithms seem sensitive to neural architectures of latent variables. It turns out latent variables in amortized policies can lead to task-specific optimal values (Schaul et al., 2015) in actor-critic policy search, but appropriate embeddings are also decisive.

5.6 CONCLUSION & DISCUSSION

Technical Discussions. We have proposed a novel meta dynamics model (GSSM), which consists of a local latent variable z_t for individual dynamics prediction and a global latent variable z_c to summarize an MDP for the policy to condition. Learning the representations of these latent variables is achieved via message passing. GSSM demonstrates its effectiveness in capturing task-specific system dynamics and exhibits good generalization capability across tasks. Meanwhile, amortized policies are developed for a more efficient meta model-based policy search. These policies allow for fast adaptation to meta-testing tasks without additional policy gradient updates and show competitive performance. It is important to note that this trait helps us avoid either re-planning or adaptation in policies. So our proposed amortized policy search is more suitable to apply in time-sensitive decision-making missions.

Existing Limitations. In principle, we summarize two limitations in this work: Even though MBMRL can theoretically achieve data efficiency and fast skill transfer, the instability of policy performance, including GSSM and other models, is unavoidable due to the bias of dynamics models. Dyna-style is used in training GSSM, so a potential limitation can be the additional time required to configure appropriate hyper-parameters, which decide when to collect new transitions and where to stop policy optimization.

Future Extensions. Designing heuristic rules are worthwhile to investigate for an optimal parameter setting of Dyna-style training. Meanwhile, we stress the importance of connections between posterior sampling and model-based meta reinforcement learning. The optimism in the face of uncertainty and fast policy adaptation can be further explored in a model-based setup.

1 A paired-t test between GSSM and MLSM-v1 over all tasks is executed, and results are significant with a default significance level 0.05.

6

BRIDGE THE INFERENCE GAPS IN NPS WITH EXPECTATION MAXIMIZATION

As revealed in the previous chapters (3)/(4/(5), solving functional representation problems in different scenarios strongly rely on diverse structure inductive biases to achieve outstanding performance. Variational inference works as the essential tool to optimize developed models. This chapter is distinguished from all the above chapters, and we study optimization objectives and inference ways within the vanilla neural process structure. We optimize the neural process with a variational expectation maximization framework, and the resulting model has the potential to obtain at least the local optimal functional representation.

6.1 INTRODUCTION

The combination of deep neural networks and stochastic processes is a powerful framework for modeling data points with correlations. As a fundamental model in our studies, the neural processes (NP) (Garnelo et al., 2018b) exploits high capacity in neural networks and enables uncertainty quantification in distribution over functions. This family of models can significantly reduce the run-time complexity of predictive distributions compared to Gaussian processes (Titsias, 2009).

Still, we study the vanilla NP as a deep directed latent variable model shown in Fig. (20). In particular, let us recap the inference methods used in vanilla NPs: It approximates the functional posterior $q_\phi(z) \approx p(z|\mathcal{D}^T;\vartheta)$ and a functional prior $q_\phi(z|\mathcal{D}^C) \approx p(z|\mathcal{D}^C;\vartheta)$, which are permutation invariant to the order of data points. Then the predictive distribution for a data point $[x_*.y_*]$ can be formulated in the form $\mathbb{E}_{q_\phi(z|\mathcal{D}^C)}\left[p(y_*|[x_*,z];\vartheta)\right]$.

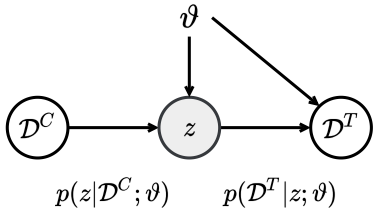


Figure 20: Deep Directed Latent Variable Models for Neural Processes. Here z is a global latent variable to summarize function properties. The model involves a functional prior distribution $p(z|\mathcal{D}^C;\vartheta)$ and a functional generative distribution $p(\mathcal{D}^T|z;\vartheta)$. Please refer to Section (6.3) for the meaning of notations.

With various structural inductive biases incorporated in modeling, variants of NPs exhibit better expressiveness and are able to handle complicated functional representation tasks, such as uncertainty-aware image classification (Wang and Van Hoof, 2020), few shot regression (Wang and Van Hoof, 2022a), and data-efficient control (Wang and Van Hoof, 2022b).

Research Motivations. Previous works (Garnelo et al., 2018a,b; Kim et al., 2019a; Gordon et al., 2019; Wang and Van Hoof, 2020, 2022a) fail to explain why the vanilla NPs suffer underfitting and less precise uncertainty quantification issues and what kind of functional priors the vanilla NPs can obtain. Meanwhile, we point out the remaining crucial issues that have not been investigated in this domain, respectively: (i) understanding the inference suboptimality of NPs (ii) quantifying statistical traits of NPs. To this end, we make more efforts to diagnose the vanilla NP from its optimization objective. Our primary interest is to find a tractable way to optimize NPs and examine the statistics of learned functional priors from diverse optimization objectives. For these issues, we make two essential claims throughout the investigations.

- **Optimization Gap.** The primary source of suboptimality of NPs originates from the ill-posed optimization objective. Due to the consistent regularizer, the optimization concerning NPs cannot guarantee the performance improvement to the likelihood of meta dataset.
- **Prior Inference.** The functional prior deserves more attention than the functional posterior since it is closely related to predictive distributions in NPs family. Hence, direct optimization concerning the prior is more effective than variational inference towards the posterior.

Developed Methods. To understand the inference suboptimality of vanilla NPs, we establish connections among a collection of optimization objectives, *e.g.* approximate evidence lower bounds (ELBOs) and Monte Carlo estimates of log-likelihoods, in Section (6.4). Then we formulate a tractable optimization objective within the variational expectation maximization framework and obtain the Self-normalized Importance weighted neural process (SI-NP) in Section (6.5).

Our primary contributions are three-fold: (i) we analyze the inherent inference suboptimality of NPs from an objective optimization perspective; (ii) we demonstrate the equivalence of conditional NPs (Garnelo et al., 2018a) and SI-NPs with one Monte Carlo sample, which closely relates to the prior collapse issue; (iii) our developed SI-NPs have an improvement guarantee to the likelihood of meta dataset in optimization and show a significant advantage over baselines with other objectives.

6.2 RELATED WORK

Structural Inductive Biases in NPs. NPs are simple and flexible in model formulations, but they suffer a bit from underfitting (Garnelo et al., 2018b; Kim et al., 2018). Previous researchers concentrate more on the use of structural inductive biases in NPs. (Kim

et al., 2019a, 2021) improves the predictive performance by adding attention based local variables. Translation equivariance and invariance are incorporated in modeling NPs with help of convolutions (Gordon et al., 2019; Foong et al., 2020; Kawano et al., 2020). To find more compact functional representations, the contrastive loss is used to regularize (C)NPs' training (Gondal et al., 2021). Hierarchical and mixture structures are also explored to induce diverse latent variables in NPs (Wang and Van Hoof, 2020, 2022a). Lee et al. (2020a) combines boostrapping tricks with neural processes to improve expressiveness of latent variables.

Expectation Maximization & Wake-Sleep Algorithms. For log-likelihood maximization problems with incomplete observations, expectation maximization (Bishop and Nasrabadi, 2006) is a tractable optimization method for latent variable models. It consists of an `E`-step to optimize the distribution of latent variables and a `M`-step to maximize the likelihood parameters. The optimization of VAEs and variants is also built upon an EM framework (Ghojogh et al., 2021; Gao et al., 2021). Our developed algorithm can be interpreted as EM for NPs. Another family of algorithms related to ours is wake-sleep algorithms (Bornschein and Bengio, 2014; Eslami et al., 2016; Le et al., 2020), where self-normalized importance sampling is used in Monte Carlo estimates. In this case, our method can be also viewed as the extension of the reweighted wake-sleep (RWS) algorithm to NPs with an improvement guarantee. Another difference between original RWS algorithm and our method lies in that optimizing a learnable functional prior is of most importance in NPs, while RWS algorithms are mostly used in scenarios with a fixed prior.

6.3 PRELIMINARIES

General Notations. We study NPs in a meta learning setup. $\mathcal{T}$ defines a set of tasks with τ a sampled task. Let $\mathcal{D}_\tau^C = \{(x_i, y_i)\}_{i=1}^n$ and $\mathcal{D}_\tau^T = \{(x_i, y_i)\}_{i=1}^{n+m}$ denote the context points for the functional prior inference and the target points for the function prediction. The latent variable z is a functional representation for a task τ. We refer to $\vartheta \in \Theta$ as the parameters of the deep directed latent variable model for NPs.

In detail, ϑ consists of encoder parameters in a functional prior $p(z|\mathcal{D}_\tau^C;\vartheta)$ and decoder parameters in a generative distribution $p(\mathcal{D}_\tau^T|z;\vartheta)$. ϕ is the parameter of an approximate posterior $q_\phi(z) = q_\phi(z|\mathcal{D}_\tau^T)$ in variational inference, while η is the parameter of a proposal distribution $q_\eta(z)$ in self-normalized importance sampling. Gaussian distributions are used as the default for these distributions, *e.g.* $p(z|\mathcal{D}_\tau^C;\vartheta) = \mathcal{N}(z;\mu_\vartheta(\mathcal{D}_\tau^C),\Sigma_\vartheta(\mathcal{D}_\tau^C))$, $q_\phi(z) = \mathcal{N}(z;\mu_\phi(\mathcal{D}_\tau^T),\Sigma_\phi(\mathcal{D}_\tau^T))$ and $q_\eta(z) = \mathcal{N}(z;\mu_\eta(\mathcal{D}_\tau^T),\Sigma_\eta(\mathcal{D}_\tau^T))$.

NPs in Meta Learning Tasks. Given a collection of tasks $\mathcal{T}$, we can decompose the marginal distribution $p(\mathcal{D}_\mathcal{T}^T|\mathcal{D}_\mathcal{T}^C;\vartheta)$ with a global latent variable z in Eq. (6.1). Here the conditional distribution $p(z|\mathcal{D}_\tau^C;\vartheta)$ with $\tau \in \mathcal{T}$ is permutation invariant *w.r.t.* the order of data points and encodes the functional prior in the generative process.

$$\underbrace{p(\mathcal{D}_\mathcal{T}^T|\mathcal{D}_\mathcal{T}^C;\vartheta)}_{\text{Marginal Likelihood}} = \prod_{\tau\in\mathcal{T}}\left[\int \underbrace{p(\mathcal{D}_\tau^T|z;\vartheta)}_{\text{Generative Likelihood}} \underbrace{p(z|\mathcal{D}_\tau^C;\vartheta)}_{\text{Functional Prior}} dz\right] \tag{6.1}$$

Throughout the chapter, the optimization objective of our interest is the marginal log-likelihood of a meta learning dataset in Eq. (6.2). Furthermore, this applies to all NP variants.

$$\max_{\vartheta} \sum_{\tau \in \mathcal{T}} \ln \left[\int p(\mathcal{D}_\tau^T | z; \vartheta) p(z | \mathcal{D}_\tau^C; \vartheta) dz \right] \tag{6.2}$$

For the sake of simplicity, we consider the task τ to derive equations in the following section[1], which corresponds to maximizing the objective

$$\mathcal{L}(\vartheta) = \ln \left[\int p(\mathcal{D}_\tau^T | z; \vartheta) p(z | \mathcal{D}_\tau^C; \vartheta) dz \right].$$

The target data points are conditionally independent given the global latent variable $p(\mathcal{D}_\tau^T | z; \vartheta) = \prod_{i=1}^{n+m} p(y_i | [x_i, z]; \vartheta)$. The marginal distribution can be interpreted as the infinite mixture of distributions when z is Gaussian. Now learning distributions over functions is reduced to a probabilistic inference problem (Garnelo et al., 2018a; Gordon et al., 2018; Kim et al., 2019a).

6.4 OPTIMIZATION GAPS AND STATISTICAL TRAITS

6.4.1 *Inference Suboptimality in vanilla NPs*

Previously, variational auto-encoder (VAE) models (Kingma and Welling, 2013; Rezende et al., 2014) mostly set a prior distribution fixed, *e.g.* $\mathcal{N}(0, I)$, as the default to approximate the posterior. This differs significantly from NPs family settings. On the one hand, the functional prior is learned in NPs. On the other hand, the functional prior participates in the performance evaluation.

Exact ELBO for NPs.

Following the essential variational inference operation, we can establish connections between the exact ELBO and the log-likelihood in Eq. (6.3). Given the functional prior $p(z | \mathcal{D}_\tau^C; \vartheta)$ and the generative distribution $p(\mathcal{D}_\tau^T | z; \vartheta)$, the exact functional posterior can be obtained by the Bayes rule

$$p(z | \mathcal{D}_\tau^T; \vartheta) = \frac{p(\mathcal{D}_\tau^T | z; \vartheta) p(z | \mathcal{D}_\tau^C; \vartheta)}{\int p(\mathcal{D}_\tau^T | z; \vartheta) p(z | \mathcal{D}_\tau^C; \vartheta) dz}.$$

The existence of the denominator $p(D_\tau^T | D_\tau^C)$ makes exact inference infeasible, and the family of variational posteriors is introduced to approximate $p(z | \mathcal{D}_\tau^T; \vartheta)$. Here the variational posterior family is defined in a parameterized set $Q_\Phi = \{q_\phi(z) | \phi \in \Phi\}$.

1 In experimental sections, both training and testing phases are implemented in a collection of tasks.

$$\mathcal{L}(\vartheta) = \ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta) = \underbrace{\mathbb{E}_{q_\phi(z)}\left[\ln \frac{p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta)}{q_\phi(z)}\right]}_{\text{Exact ELBO}} + \underbrace{D_{KL}\left[q_\phi(z) \parallel p(z|\mathcal{D}_\tau^T;\vartheta)\right]}_{\text{Posterior Approximation Gap}} \tag{6.3}$$

When the variational posterior family is flexible enough, *e.g.* $p(z|\mathcal{D}_\tau^T;\vartheta) \in \mathcal{Q}_\Phi$, the posterior approximation gap can be reduced to a tiny quantity. In this case, maximizing the exact ELBO in Eq. (6.4) brings the increase of the likelihood in Eq. (6.3) accordingly.

$$\mathcal{L}_{\text{ELBO}}(\vartheta,\phi) = \mathbb{E}_{q_\phi(z)}\left[\ln p(\mathcal{D}_\tau^T|z;\vartheta)\right] - D_{KL}\left[q_\phi(z) \parallel p(z|\mathcal{D}_\tau^C;\vartheta)\right] \tag{6.4}$$

Approximate ELBO for NPs.

As previously mentioned, the inference is complicated since the functional prior and the posterior in the exact ELBO are unknown. To this end, Garnelo et al. (2018b) proposes a surrogate objective as an approximate ELBO for NPs. This is defined as Eq. (6.5) to maximize.

$$\mathcal{L}_{\text{NP}}(\vartheta,\phi) = \mathbb{E}_{q_\phi(z)}\left[\ln \underbrace{p(\mathcal{D}_\tau^T|z;\vartheta)}_{\text{Generative Likelihood}}\right] - \underbrace{D_{KL}\left[q_\phi(z) \parallel q_\phi(z|\mathcal{D}_\tau^C)\right]}_{\text{Consistent Regularizer}} \tag{6.5}$$

Here the approximate posterior $q_\phi(z)$ and the approximate prior $q_\phi(z|\mathcal{D}_\tau^C)$ are constrained in a Kullback-Leibler divergence term, referred to as the consistent regularizer in this chapter. We claim that the consistent regularizer is the source of the suboptimality of vanilla NPs, and this is shown in Appendix (D.3) as the proof of Remark (1).

Remark 1 *The Eq. (6.5) is a problematic objective w.r.t. maximizing the likelihood of the dataset.*

Other Available Objectives in NPs Family.

Now we turn to other tractable optimization objectives in NPs. These include conditional neural processes (CNPs) (Garnelo et al., 2018a) and convolutional neural processes (ConvNPs) (Foong et al., 2020).

Here the CNP is treated as a particular case of NPs. The objective $\mathcal{L}_{\text{CNP}}(\vartheta)$ can be obtained when the functional prior collapses into a Dirac delta distribution.

$$\mathcal{L}_{\text{CNP}}(\vartheta) = \mathbb{E}_{p(z|\mathcal{D}_\tau^C;\vartheta)}\left[\ln p(\mathcal{D}_\tau^T|z;\vartheta)\right] \quad \text{with} \quad p(z|\mathcal{D}_\tau^C;\vartheta) = \delta(|z-\hat{z}|) \tag{6.6}$$

For the ConvNP, we skip the convolutional structural inductive bias and concentrate more on the optimization objective itself. Its objective in Eq. (6.7) is a biased Monte

Carlo estimate of Eq. (6.2). A benefit is that we can maximize the log-likelihood of marginal distributions straightforwardly.

$$\mathcal{L}_{\text{ML-NP}}(\vartheta) = \ln\left[\frac{1}{B}\sum_{b=1}^{B}\exp\Big(\ln p(\mathcal{D}_\tau^T|z^{(b)};\vartheta)\Big)\right] \quad \text{with} \quad z^{(b)} \sim p(z|\mathcal{D}_\tau^C;\vartheta) \tag{6.7}$$

This is termed as Monte Carlo Maximum Likelihood $\mathcal{L}_{\text{ML-NP}}(\vartheta)$[2], and B is the number of used Monte Carlo samples. Without the involvement of the consistent regularizer, both $\mathcal{L}_{\text{CNP}}(\vartheta)$ and $\mathcal{L}_{\text{ML-NP}}(\vartheta)$ will not encounter the approximation gap in practice.

6.4.2 *Evaluation Criteria & Asymptotic Performance*

As in (Le et al., 2018; Foong et al., 2020), we take a multi-sample Monte Carlo method to evaluate the performance. This applies to both meta training and meta testing processes. In detail, NP models need to run B times stochastic forward pass by sampling $z^{(b)} \sim p(z|\mathcal{D}_\tau^C;\vartheta)$ and then compute the log-likelihoods as $\ln\Big[\frac{1}{B}\sum_{b=1}^{B} p(\mathcal{D}_\tau^T|z^{(b)};\vartheta)\Big]$.

After describing the evaluation criteria, we turn to a phenomenon of our interest. In Gaussian processes (Ghahramani, 2015), with more observed context points, the epistemic uncertainty can be decreased, and the predictive mean function is closer to the ground truth.

Similarly, this trait is also reflected in NPs family and can be quantitatively described as follows. Given metrics of an average predictive error β, the number of context points n and the evaluated dataset, $\mathcal{D}_\tau^T$, the introduced metrics $\beta(\mathcal{D}_\tau^T;n)$ are decreased when increasing n in prediction. In our paper, we refer to this trait as the asymptotic effect.

Definition 2 (Prior Collapse) *The functional prior $p(z|\mathcal{D}_\tau^C;\vartheta) = \mathcal{N}(z;\mu_\vartheta(\mathcal{D}_\tau^C),\Sigma_\vartheta(\mathcal{D}_\tau^C))$ is said to collapse in learning when the trace of the covariance matrix satisfies*

$$Tr[\Sigma_\vartheta(\mathcal{D}_\tau^C)] = \sum_{i=1}^{d}\sigma_i^2 \approx 0$$

with $\Sigma_\vartheta(\mathcal{D}_\tau^C) = diag[\sigma_1^2,\ldots,\sigma_d^2]$.

As previously mentioned, we can more precisely keep track of measures $\beta(\mathcal{D}_\tau^T;n)$, such as predictive log-likelihoods or mean square errors of data points, to assess the asymptotic behavior. The role of latent variables z is to propagate the uncertainty about the partial observations in functions. And **Definition** (2) provides a quantitative way to examine the extent of prior collapse in ML-NPs and SI-NPs.

6.5 TRACTABLE OPTIMIZATION VIA EXPECTATION MAXIMIZATION

In this section, we propose alleviating the inference suboptimality of NPs by formulating a surrogate optimization objective. Meanwhile, we design the variational expectation

2 The Monte Carlo maximum likelihood corresponds to the optimization objective that in (Foong et al., 2020) except that the convolutional inductive bias is removed.

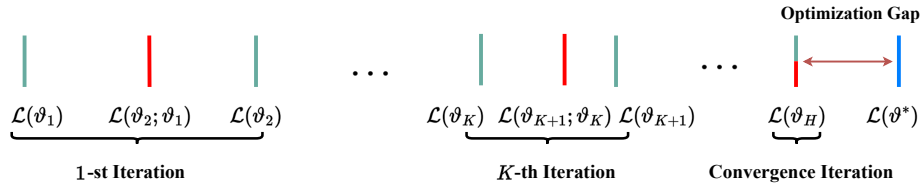


Figure 21: Illustration of Expectation Maximization for NPs. Green lines indicate the results after the E-steps while the red lines are for the M-steps in Algorithm (8). In the convergence iteration, the performance gap $\mathcal{L}(\vartheta_H) - \mathcal{L}(\vartheta_{H-1})$ is close to zero and the algorithm results in at least a local optimal solution. Values of these quantities are increased from the left to the right.

maximization algorithm to accommodate this objective. The benefit of our method is to guarantee performance improvement *w.r.t.* the likelihood of meta dataset in iterations and finally result in at least a local optimal.

6.5.1 *Variational Expectation Maximization for NPs*

This part is to bridge the inference suboptimality as mentioned earlier. We retain the neural architectures used in NPs. The basic idea has already been illustrated in Fig. (21).

In detail, we iteratively construct the lower bound $\mathcal{L}(\vartheta_K)$ and maximize the surrogate function $\mathcal{L}(\vartheta_{K+1};\vartheta_K)$. The referred optimization gap is sourced from the optimizer and measures the difference between converged (local) optimal functional prior and the theoretical optimal functional prior. The general pseudo code is Algorithm (8).

Algorithm 8: Variational Expectation Maximization for NPs.

Input : Task distribution $p(\mathcal{T})$; Task batch size, Number of particles, Initialized ϑ and η.

Output : Meta-trained parameters ϑ and η.

for $k = 1$ ***to*** K **do**
  E-step #1: $k \leftarrow k+1$ and reset the variational posterior $q_\phi(z) = p(z|\mathcal{D}_\tau^T;\vartheta_k)$ in Eq. (6.3);
  if *Use the Conditional Prior as the Proposal* **then**
    Set $q_\eta(z|\mathcal{D}_\tau^T) = p(z|\mathcal{D}_\tau^C;\vartheta_k)$;
  else
    E-step #2: update the proposal $\eta_k = \arg\min_\eta \mathcal{L}_{\text{KL}}(\eta;\eta_{k-1},\vartheta_k)$ in Eq. (6.13);
  end
  M-step: optimize surrogate functions $\vartheta_{k+1} = \arg\max_\vartheta \mathcal{L}_{\text{SI-NP}}(\vartheta;\eta_k,\vartheta_k)$ in Eq. (6.11);
end

Surrogate Function for Exact NPs

Definition 3 (Surrogate Function) [3] *Given the objective function $f(\vartheta)$ to maximize, surrogate functions $g(\vartheta;\vartheta_k)$ w.r.t. $f(\vartheta)$ are a family of functions with the following properties.*

- *Both $f(\vartheta)$ and $g(\vartheta;\vartheta_k)$ are C^2-functions, which means $f(\vartheta)$ has first order and second order derivatives of those functions exist and these are smooth in the domain of the function.*
- *The following two formulas hold:*

$$g(\vartheta;\vartheta_k) \leq f(\vartheta)\ \forall\vartheta; \quad g(\vartheta_k;\vartheta_k) = f(\vartheta_k). \tag{6.8}$$

To make the optimization of meta dataset log-likelihood feasible, we construct surrogate functions as the proxy in each iteration step. These meta learning surrogate functions with special properties are closely connected with the original objective, and we leave this discussion in Appendix (D.5).

Here ϑ_k denotes the parameter of directed latent variable models for NPs in the k-th iteration of variational expectation maximization. In Eq. (6.4), we take the step by replacing the approximate posterior with the last time updated $p(z|\mathcal{D}_\tau^T;\vartheta_k)$ in Algorithm (8). And this results in the following equation,

$$\mathcal{L}(\vartheta;\vartheta_k) = \mathbb{E}_{p(z|\mathcal{D}_\tau^T;\vartheta_k)}\left[\ln p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta) - \ln p(z|\mathcal{D}_\tau^T;\vartheta_k)\right] \tag{6.9}$$

where $p(z|\mathcal{D}_\tau^T;\vartheta_k)$ is the posterior distribution.

Proposition 3 *The proposed meta learning function $\mathcal{L}(\vartheta;\vartheta_k)$ in Eq. (6.9) is a surrogate function w.r.t. the log-likelihood of the meta learning dataset.*

The above proposition is examined based on the definition in Appendix (D.5.2).

Tractable Optimization with Self-Normalized Importance Sampling

Since the second term in Eq. (6.9) is constant in the iteration, we can drop it to simplify the surrogate objective as the right side of the following equation.

$$\max_\vartheta \mathcal{L}(\vartheta;\vartheta_k) \Leftrightarrow \max_\vartheta \mathcal{L}_{\text{EM}}(\vartheta;\vartheta_k) = \mathbb{E}_{p(z|\mathcal{D}_\tau^T;\vartheta_k)}\left[\ln p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta)\right] \tag{6.10}$$

Proposition 4 *Optimizing this surrogate function of a batch of tasks via the variational expectation maximization leads to an improvement guarantee w.r.t. $\sum_{\tau\in\mathcal{T}} \ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta)$.*

3 Note that the surrogate function above is defined in the Minorize-Maximization (MM) algorithm (Hunter and Lange, 2004)

Still we cannot optimize $\mathcal{L}_{\text{EM}}(\vartheta;\vartheta_k)$ since the expected form is not analytical and is intractable to sample from $p(z|\mathcal{D}_\tau^T;\vartheta_k)$ for Monte Carlo estimates[4]. Remember that the marginal distribution $p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_k)$ is task dependent and can not be ignored in computing the posterior $p(z|\mathcal{D}_\tau^T;\vartheta_k)$. To circumvent this, we introduce a proposal distribution $q_\eta(z|\mathcal{D}_\tau^T)$ and optimize the objective via self-normalized importance sampling (Tokdar and Kass, 2010). The resulting meta learning surrogate function is as follows.

$$\begin{aligned}\mathcal{L}_{\text{EM}}(\vartheta;\vartheta_k) &= \mathbb{E}_{q_\eta}\left[\frac{p(z|\mathcal{D}_\tau^T;\vartheta_k)}{q_\eta(z|\mathcal{D}_\tau^T)}\ln p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta)\right] \\ &\approx \sum_{b=1}^{B}\hat{\omega}^{(b)}\ln p(\mathcal{D}_\tau^T, z^{(b)}|\mathcal{D}_\tau^C;\vartheta) \\ &= \sum_{b=1}^{B}\underbrace{\hat{\omega}^{(b)}}_{\text{Importance Weight}}\left[\ln\underbrace{p(\mathcal{D}_\tau^T|z^{(b)};\vartheta)}_{\text{Generative Likelihood}} + \ln\underbrace{p(z^{(b)}|\mathcal{D}_\tau^C;\vartheta)}_{\text{Functional Prior Likelihood}}\right] \\ &= \mathcal{L}_{\text{SI-NP}}(\vartheta;\eta_k,\vartheta_k)\end{aligned} \tag{6.11}$$

where $z^{(b)} \sim q_{\eta_k}(z|\mathcal{D}_\tau^T)$,

$$\omega^{(b)} = \exp\left(\ln p(\mathcal{D}_\tau^T|z^{(b)};\vartheta_k) + \ln p(z^{(b)}|\mathcal{D}_\tau^C;\vartheta_k) - \ln q_{\eta_k}(z^{(b)}|\mathcal{D}_\tau^T)\right)$$

and $\hat{\omega}^{(b)} = \frac{\omega^{(b)}}{\sum_{b'=1}^{B}\omega^{(b')}}$.

In terms of the first conditional term in Eq. (6.11), all the data points are conditional independent and this is further expressed as $\ln p(\mathcal{D}_\tau^T|z^{(b)};\vartheta) = \sum_{i=1}^{n+m}\ln p(y_i|[x_i, z^{(b)}];\vartheta)$.

Proposition 5 *With one Monte Carlo sample used in Eq. (6.11), the presumed diagonal Gaussian prior $p(z|\mathcal{D}_\tau^C;\vartheta)$ will collapse into a Dirac delta distribution in convergence. In this case, the one sample Monte Carlo approximate objective of SI-NP in Eq. (6.12) is equivalent with CNP in Eq. (6.6).*

$$\mathcal{L}_{\textit{SI-NP}}(\vartheta;\eta_k,\vartheta_k) \approx \mathbb{E}_{p(z|\mathcal{D}_\tau^C;\vartheta_k)}\left[\ln\underbrace{p(\mathcal{D}_\tau^T|z;\vartheta)}_{\textit{Generative Likelihood}}\right] + \underbrace{\mathbb{E}_{p(z|\mathcal{D}_\tau^C;\vartheta_k)}\left[\ln p(z|\mathcal{D}_\tau^C;\vartheta)\right]}_{\textit{Prior Collapse Term}} \tag{6.12}$$

The **Proposition** (5) establishes connections between SI-NPs and CNPs in optimization and we attribute this to the collapse term in Eq. (6.12). Hence, CNP can be viewed as a particular example in SI-NPs. The complete proof is based on limit analysis and can be found in Appendix (D.6).

4 Though the exact posterior distribution $p(z|\mathcal{D}^T;\vartheta)$ can be inferred by Bayes rule, the denominator $\int p(\mathcal{D}_\tau^T|z;\vartheta)p(z|\mathcal{D}_\tau^C;\vartheta)dz$ is not available.

Update of Proposal Distributions (Optional)

The use of proposal distribution q_η enables sampling z for Monte Carlo estimates. Another role of the proposal distribution is to work as a proxy for the posterior $p(z|\mathcal{D}_\tau^T;\vartheta_k)$, and the variance $\texttt{Var}_{q_\eta}\left[\frac{p(z|\mathcal{D}_\tau^T;\vartheta_k)}{q_\eta(z|\mathcal{D}_\tau^T)}\ln p(\mathcal{D}_\tau^T,z|\mathcal{D}_\tau^C;\vartheta)\right]$ is expected to be lower for stable training. So a reasonable optimization objective is to get two distributions as close as each other, *e.g.* minimizing the Kullback–Leibler divergence $D_{KL}[p(z|\mathcal{D}_\tau^T;\vartheta_k)\parallel q_\eta(z|\mathcal{D}_\tau^T)]$.

This treatment is equivalent to wake phase updates in (Bornschein and Bengio, 2014). We can obtain the optimization objective on the right side of Eq. (6.13) with the help of the self-normalized importance sampling.

$$\begin{aligned}&\min_\eta D_{KL}[p(z|\mathcal{D}_\tau^T;\vartheta_k)\parallel q_\eta(z|\mathcal{D}_\tau^T)]\\ \Leftrightarrow \min_\eta \mathcal{L}_{\mathrm{KL}}(\eta;\eta_{k-1},\vartheta_k) &= -\sum_{b=1}^{B}\hat{\omega}^{(b)}\ln q_\eta(z^{(b)}|\mathcal{D}_\tau^T)\end{aligned} \tag{6.13}$$

Here the self-normalized importance weights $\{\hat{\omega}^{(b)}\}_{b=1}^B$ inside the above equation are the same with that in Eq. (6.11). Also note that the the denominator $\frac{1}{B}\sum_{b'=1}^B\omega^{(b')}$ of the weight relates to the estimate of $p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C)$, which has biases with limited samples at the beginning of training. But with the improvement of approximation, the bias can be decreased accordingly (Zimmermann et al., 2021).

In practice, the selection of proposal distributions can be tricky. We make the update of proposal distributions optional in implementations. In our experimental settings, we simply use the conditional prior $p(z|\mathcal{D}_\tau^C;\vartheta)$ as the default, which is competitive enough in performance.

6.5.2 *Scalable Training and Testing*

As shown in Algorithm (8), the meta training process consists of two steps. As it turns out to be challenging to stably optimize E-step #2, we skip this optional step. Instead, we use $p(z|\mathcal{D}_\tau^C;\vartheta)$ as the proposal distribution. By repeating E-step #1/M-step iterations until convergence, the method can theoretically find at least a local optimal *w.r.t.* the log-likelihood of meta learning dataset. The previous **Proposition** (4) guarantees this property.

Once the learning progress reaches the final convergence, we can make use of the learned functional prior to obtain the predictive distribution. With B particles in prediction, the distribution can be expressed as follows.

$$\begin{aligned}p(y|x,\mathcal{D}_\tau^C;\vartheta) &= \mathbb{E}_{p(z|\mathcal{D}_\tau^C;\vartheta)}p(y|[x,z];\vartheta)\\ \approx \frac{1}{B}\sum_{b=1}^{B}p(y|[x,z^{(b)}];\vartheta) &\text{ with } z^{(b)}\sim p(z|\mathcal{D}_\tau^C;\vartheta)\end{aligned} \tag{6.14}$$

6.6 EXPERIMENTS

In this section, two central questions are answered: (i) can variational EM based models SI-NPs, lead to better local optimal? (ii) what is the role of randomness in functional priors? Specifically, we examine the influence of NPs optimization objectives on typical downstream tasks and understand the functional prior quantitatively.

Baselines & Evaluations. Since our concentration is on optimization objectives in NPs family, we simply include NP (Garnelo et al., 2018b), and CNP (Garnelo et al., 2018a), ML-NP (Foong et al., 2020) in comparison. Note that our developed SI-NP and ML-NP (Foong et al., 2020) are importance weighted models, but the mechanisms in estimating weights are significantly different. As for evaluations, we refer the reader to Sec. (6.4.2) for more information.

6.6.1 *Synthetic Regression*

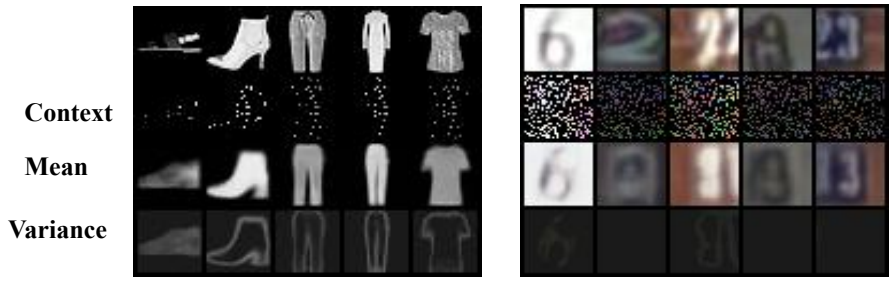
We create synthetic regression tasks by sampling functions from Gaussian processes. Three types of kernels, respectively Matern$-\frac{5}{2}$, RBF, and Periodic, are used to generate diverse function distributions. In each iteration, a batch of data points from functions is randomly processed into the context and the target for training models. In meta testing, the same strategy is used to generate functions and data points. For each kernel case, we sample 1000 tasks to evaluate the average log-likelihoods and standard deviations. We report the results in Table (10). It shows SI-NPs outperform the vanilla NPs in all cases and are at least competitive with other baselines. Though ML-NPs are superior to NPs, they cannot beat CNPs in RBF and Periodic cases.

Table 10: Test average log-likelihoods with reported standard deviations for 1-dimensional Gaussian process dataset with various kernels.

#	Matern $-\frac{5}{2}$	RBF	Periodic
$\mathcal{L}_{\text{NP}}$ (Garnelo et al., 2018b)	-0.225±0.03	-0.183±0.03	-0.611±0.034
$\mathcal{L}_{\text{CNP}}$ (Garnelo et al., 2018a)	0.295±0.017	0.463±0.023	-0.533±0.009
$\mathcal{L}_{\text{ML-NP}}$ (Foong et al., 2020)	0.303±0.013	0.439±0.009	-0.547±0.036
$\mathcal{L}_{\text{SI-NP}}$ (ours)	0.305±0.006	0.493±0.007	-0.532±0.036

6.6.2 *Image Completion*

Similar to experiments in (Garnelo et al., 2018b; Kim et al., 2019a), we perform image completion experiments in this section. We implement NPs baselines in four commonly used datasets, namely MNIST (Bottou et al., 1994), FMNIST (Xiao et al., 2017), CIFAR10 (Krizhevsky et al., 2009) and SVHN (Sermanet et al., 2012). In meta training processes, the mission is to randomly mask the pixels and then try to complete the images with observed pixels. In detail, we learn latent variable models that map all pixel locations $x \in [0, 1]^2$ to Gaussian distribution parameters of pixel values based on the



(a) (b)

Figure 22: Examples of Completed Images using SI-NP. We randomly sample test images from FMNIST and SVHN datasets, and ground truth images are in the first row. The predictive means and variances of completed images are visualized in the third and fourth rows.

context pixel locations and values. Fig. (22) is an example of completion results from sampled images. For implementation details, please refer to Appendix (D.4).

Table 11: Test average log-likelihoods with reported standard deviations for image completion in MNIST/FMNIST/SVHN/CIFAR10 (5 runs). We test the performance of different optimization objectives in both context data points and target data points. Except CNPs, we use 32 Monte Carlo samples from the functional prior to evaluate the average log-likelihoods. Bold ones are the best.

	MNIST		FMNIST		SVHN		CIFAR10	
#	context	target	context	target	context	target	context	target
$\mathcal{L}_{\text{NP}}$	$0.81_{\pm 0.006}$	$0.73_{\pm 0.007}$	$0.83_{\pm 0.007}$	$0.73_{\pm 0.009}$	$3.19_{\pm 0.02}$	$3.07_{\pm 0.02}$	$2.35_{\pm 0.04}$	$2.03_{\pm 0.02}$
$\mathcal{L}_{\text{CNP}}$	$1.05_{\pm 0.005}$	$0.99_{\pm 0.008}$	$0.95_{\pm 0.007}$	$0.90_{\pm 0.009}$	$\mathbf{3.57}_{\pm 0.003}$	$3.48_{\pm 0.004}$	$2.71_{\pm 0.004}$	$2.53_{\pm 0.006}$
$\mathcal{L}_{\text{ML-NP}}$	$1.06_{\pm 0.004}$	$0.99_{\pm 0.006}$	$0.94_{\pm 0.008}$	$0.89_{\pm 0.007}$	$3.51_{\pm 0.008}$	$3.43_{\pm 0.006}$	$2.60_{\pm 0.005}$	$2.41_{\pm 0.005}$
$\mathcal{L}_{\text{SI-NP}}$ (ours)	$\mathbf{1.09}_{\pm 0.006}$	$\mathbf{1.02}_{\pm 0.004}$	$\mathbf{0.98}_{\pm 0.004}$	$\mathbf{0.94}_{\pm 0.005}$	$\mathbf{3.57}_{\pm 0.003}$	$\mathbf{3.50}_{\pm 0.003}$	$\mathbf{2.75}_{\pm 0.004}$	$\mathbf{2.60}_{\pm 0.005}$

We examine the performance in a testing subset of the above image datasets and report the average log-likelihoods in Table (11). Note that in evaluation, a random number of pixels is selected to formulate the context points for each image, and the Table (11) is the mean result. We can see that SI-NP achieves the best performance in all image datasets. Significantly, performance differences between SI-NPs and NPs are primarily huge. Furthermore, the asymptotic behaviors of all baselines are illustrated in Fig. (23). All models exhibit asymptotic behaviors by varying the number of context points, and NPs mainly cause extremely lower log-likelihoods with 10 context points. The observations at different levels are consistent with the conclusion in Table (11).

Another critical finding lies in Fig. (24), which checks the difference between approximate posteriors and priors in NPs and examines whether the learned functional priors collapse into the deterministic ones. In MNIST dataset, SI-NPs encounter the prior collapse, and ML-NPs also obtain extremely lower trace values of covariance matrices. We attribute the prior collapse in MNIST to its most superficial structures and limited semantic diversity. For image datasets with more complicated semantics, the computed trace values in SI-NPs retain a reasonable interval. The scale of SI-

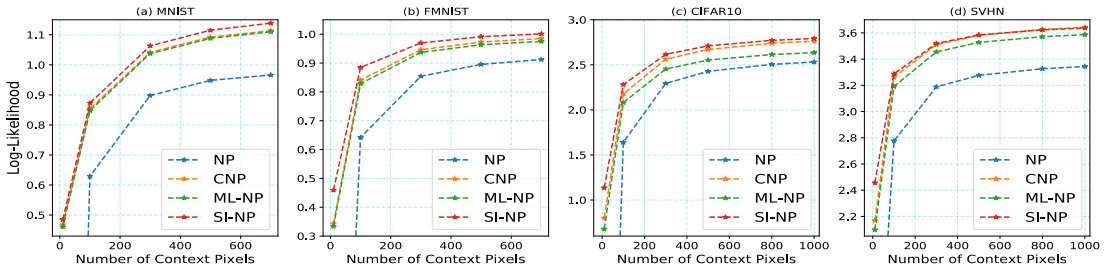


Figure 23: Asymptotic Performance in Image Completion. We meta test pixel average log-likelihoods with varying number of context points in image datasets. Context points are randomly selected for each image in testing processes. For MNIST/FMNIST datasets, the numbers of context pixels in testing are {10, 100, 300, 500, 700}. For CIFAR10/SVHN datasets, the numbers of context pixels in testing are {10, 100, 300, 500, 800, 1000}.

NPs' trace values can implicitly reveal the semantics complexity of image datasets: CIFAR10>SVHN>FMNIST>MNIST.

The above reflects the meaning of the learned functional priors in SI-NPs: with more complicated semantics, the prior distribution exhibits higher uncertainty, and hence it can generate functional samples with more diversity. Similarly, the scale of KL divergence values in vanilla NPs in the first line of Fig. (24) is positively correlated with the semantic complexity. Also, note that the approximate functional prior in vanilla NPs seldom collapses, but it has a theoretical bias away from the optimal functional prior.

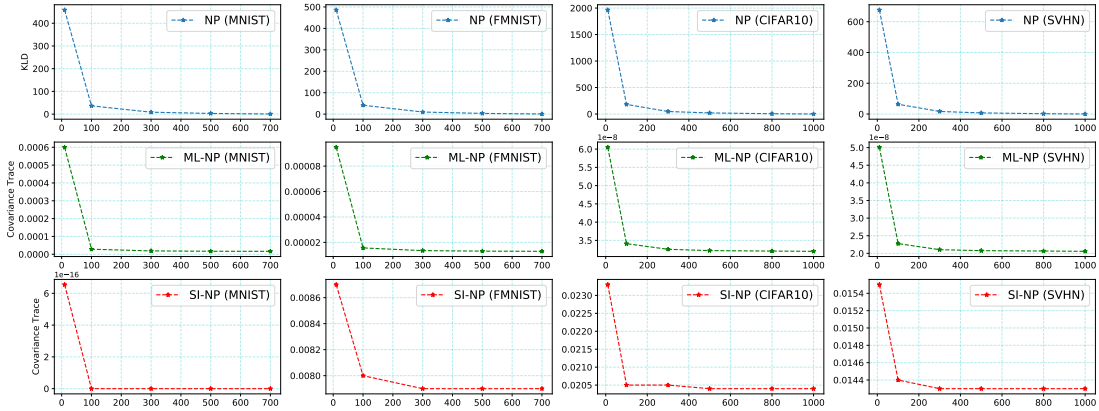


Figure 24: Statistics of Learned Functional Priors. In meta testing, we still vary the number of context points in image datasets. For NPs, the KL divergence value $D_{KL}\left[q_\phi(z) \parallel q_\phi(z|\mathcal{D}_\tau^C)\right]$ is computed. For ML-NPs/SI-NPs, the trace of learned functional priors' covariance matrices $\mathrm{Tr}[\Sigma_\vartheta(\mathcal{D}_\tau^C)]$ is computed with $p(z|\mathcal{D}_\tau^C;\vartheta) = \mathcal{N}(z;\mu_\vartheta(\mathcal{D}_\tau^C),\Sigma_\vartheta(\mathcal{D}_\tau^C))$.

6.7 CONCLUSION & DISCUSSION

Technical Discussions. In this chapter, we study NPs family from an optimization objective perspective and analyze the inference suboptimality of vanilla NPs. Within the variational expectation maximization framework, our developed SI-NPs are guaranteed

to obtain at least a local optimal functional prior. Besides, experimental results tell us the learned functional prior has close connections with the number of context points and complexity of function families.

Existing Limitations. Compared to vanilla NPs, our developed SI-NP requires more than one Monte Carlo sample from the functional prior in training. This causes additional computations but can effectively avoid the prior collapse. Though intuitively, the uncertainty of the functional prior can be forward propagated to the output distribution, the influence of such uncertainty has not been mathematically studied.

Future Extensions. The objective of the SI-NP is regardless of neural architectures so that any structural inductive bias can be easily incorporated with ours. Theoretically, this combination will produce superior functional representation baselines in the domain. Another extension lies in the improvement of proposal distributions. The setup in this chapter directly uses the functional prior as the proposal to estimate the importance weights. If we can find a reasonable objective for stably training the proposal distributions, SI-NPs are likely to achieve even better performance.

7

CONCLUSION

In this thesis, we are in pursuit of effective functional representations, which can be used for uncertainty quantification and fast skill transfer. The research topic is motivated by the neural process (NP) model (Garnelo et al., 2018b), which represents the underlying function with a global latent variable. Though the modeling in NPs is simple and elegant, it fails to generalize well in some tasks and suffers suboptimal results. In order to make NPs more practical, we partition research pipelines into two parts: (i) incorporating various inductive biases in the functional representation to accommodate complicated scenarios and (ii) making diagnoses with respect to the inference suboptimality of vanilla NPs and improving the model from an optimization perspective.

For the first part, we report our developed models and discoveries in Chapters (3)/(4)/(5). We believe the inductive bias plays a crucial role in generalizing NPs to various scenarios: DSVNP (Wang and Van Hoof, 2020) suits function datasets with high dimensional inputs/outputs well, MoE-NP (Wang and Van Hoof, 2022a) can recover functions with diverse components, and GSSM+APS (Wang and Van Hoof, 2022b) learns optimal representations of value functions/policies in data efficient control. The effectiveness of the functional representations with mentioned models are examined in a wide range of benchmarks: uncertainty quantification in image classification, curve fitting in stochastic processes, physics system identification, image completion, and meta reinforcement learning.

For the second part, in Chapter (6), we start with objectives in NPs families and then build connections between evidence lower bounds (ELBOs), maximum likelihood objectives with Monte Carlo estimates, and Self-normalized Importance weighted objectives. We do not modify the neural architecture of NPs, but focus on the optimization objective. In this chapter, we theoretically show that training within the expectation maximization framework can bring a functional representation with an improvement guarantee over the likelihood of function datasets. The resulting SI-NP (Wang et al., 2022) can unify most NP models and shows significant performance advantages over typical baselines. Such discovery is quite encouraging, and SI-NP can be combined with any inductive biases for NPs, *e.g.* translation equivariance (Gordon et al., 2019), attention modules (Kim et al., 2019a), mixture of experts (Wang and Van Hoof, 2022a) and graph structures (Wang and Van Hoof, 2022b), to release the potential of functional representations further.

To conclude, we have proposed various techniques in improving the functional representation with deep latent variable models. Admittedly, learning exact functional representations is difficult. There remain a series of issues to investigate in the future:

More Inductive Biases & Advanced Inference Methods. Available inductive biases and probabilistic inference methods play different but critical roles in learning functional representations. The design and choice of structural inductive biases depend on the trait of functional representation problems. For example, spatio-temporal datasets require translation equivariance in the spatio-temporal domain, and this can inspire another inductive bias used in functional representations. Concerning advanced inference methods, it is an open question of how to avoid the functional prior collapse (refer to Chapter (6)) in the meta training process. These two core topics still deserve more attention in the future.

Understanding Uncertainty and the Functional Representation. As known in this domain, exact uncertainty quantification is a long-standing challenge for functions. In the NP family, running multiple stochastic forward passes provides a way to measure the uncertainty roughly. However, the connections between the learned variance parameters in the functional prior and the estimated variance parameters in the output distribution are not yet uncovered. At the same time, we need a general measure to quantify the diversity and realism in generated functions. Another direction worth exploring is the decomposition of uncertainty learned from the functional representation models. Fine-grained uncertainty, such as aleatoric and epistemic uncertainty, will be helpful in decision-making.

Real-World Application Scenarios. It is also necessary to explore the deployment of the functional representation models in real-world scenarios: With partially collected sequence information, there is the potential to generate the structure of molecules, which can accelerate drug discovery. By analyzing the past sensory wildfire dataset, we can develop the functional representation model to monitor wildfire situations and pose the alert in advance. We believe our life will benefit from this family of interpretable and generative models to an extent.

8

APPENDICES

This chapter provides all necessary supplementary materials for our developed methods DSVNP in Chapter (3)/MoE-NP in Chapter (4)/GSSM-APS in Chapter (5)/SI-NP in Chapter (6). These include proofs of propositions, lemmas and theorems, required neural architectures, implementation details, and external experimental results.

A SUPPLEMENTARY MATERIALS IN CHAPTER 3

A.1 *Some Basic Concepts*

At first, we revisit some basics, which would help us understand the properties of GPs and NPs. Fundamental concepts include permutation invariant function (PIF), general stochastic function process (GSFP) etc. We stress the importance of relationship disentanglement between GPs and NPs and motivations in approximating stochastic processes (SPs) in the main passage.

Definition 4 (Permutation Invariant Function) *The f mapping a set of N M-dimension elements to a D-dimension vector variable is said to be a permutation invariant function if:*

$$\begin{aligned} f : \times_{i=1}^{N} \mathcal{R}_i^M \to \mathcal{R}^D \\ x = [x_1, \ldots, x_N] \mapsto f = \left[f_1(x_{\pi(1:N)}), \ldots, f_D(x_{\pi(1:N)}) \right] \end{aligned} \tag{8.1}$$

where x_i is a M-dimensional vector, $x_{1:N} = [x_1, x_2, \ldots, x_N]$ is a set, operation $\pi : [1, 2, \ldots, N] \mapsto [\pi_1, \pi_2, \ldots, \pi_N]$ imposes a permutation over the order of elements in the set. The PIF suggests the image of the map is regardless of the element order. Another related concept permutation equivariant function (PEF) keeps the order of elements in the output consistent with that in the input under any permutation operation π.

$$\begin{aligned} f : \times_{i=1}^{N} \mathcal{R}_i^M \to \mathcal{R}^N \\ x_\pi = (x_{\pi_1}, \ldots, x_{\pi_N}) \mapsto f_\pi = \pi \circ f(x_{1:N}) \end{aligned} \tag{8.2}$$

The **Definition** (4) is important since the exchangeable stochastic function process of our interest in this domain is intrinsically a distribution over set function, and PIF plays an inductive bias in preserving invariant statistics.

Permutation invariant functions are candidate functions for learning embeddings of a set or other order uncorrelated data structure $X_{1:N}$, and several examples can be listed in the following forms.

Example 2 (Permutation Invariant Functions) *Some structure with mean/summation over the output:*

$$F(X_{\pi(1:N)}) = \Big(\frac{1}{N}\Sigma_{i=1}^{N}\phi_1(x_i), \ldots, \frac{1}{N}\Sigma_{i=1}^{N}\phi_M(x_i)\Big) \tag{8.3}$$

Example 3 (Permutation Invariant Functions) *Some structure with Max/Min/Top k-th operator over the output (Take maximum operator for example), such as:*

$$F(X_{\pi(1:N)}) = \Big(max_{i\in\{1,2,\ldots,N\}}\phi_1(x_i), \ldots, max_{i\in\{1,2,\ldots,N\}}\phi_M(x_i)\Big) \tag{8.4}$$

Example 4 (Permutation Invariant Functions) *Some structure with symmetric higher order polynomials or other functions with a symmetry bi-variate function ϕ:*

$$F(X_{\pi(1:N)}) = \Big(\Sigma_{i,j=\{1,2,\ldots,N\}}\phi_1(x_i,x_j), \ldots, \Sigma_{i,j=\{1,2,\ldots,N\}}\phi_M(x_i,x_j)\Big) \tag{8.5}$$

The invariant property is easy to be verified in these cases, and note that in all settings of NP family in this paper, Eq. (8.3) is used only. For the bi-variate symmetric function in Eq. (8.5) or other more complicated operators would result in more flexible functional translations, but additional computation is required as well. Some of the transformations mentioned above are instantiations in DeepSet. Further investigations in this domain can exploit these higher order permutation invariant neural networks into NPs since more correlations or higher order statistics in the set can be mined for prediction. Additionally, Set Transformer is believed to be powerful in a permutation invariant representation.

Definition 5 (General Stochastic Function Process) *Let $\mathcal{X}$ denote the Cartesian product of some intervals as the index set and let dimension of observations $d \in \mathbb{N}$. For each $k \in \mathbb{N}$ and any finite sequence of distinct indices $x_1, x_2, .., x_k \in \mathcal{X}$, let $\nu(x_1, x_2, .., x_k)$ be some probability measure over $(\mathbb{R}^d)^k$. Suppose the used measure satisfies Kolmogorov Extension Theorem, then there exists a probability space $(\Omega, \mathcal{F}, \mathcal{P})$, which induces a general stochastic function process (GSFP) $\mathcal{F} : \mathcal{X} \times \Omega \to \mathbb{R}^d$, keeping the property $\nu_{(x_1,x_2,..,x_k)}(C_1 \times C_2 \times ... \times C_k) = \mathcal{P}(\mathcal{F}(x_1) \in C_1, \mathcal{F}(x_2) \in C_2, ..., \mathcal{F}(x_k) \in C_k)$ for all $x_i \in \mathcal{X}$, $d \in \mathbb{N}$ and measurable sets $C_i \in \mathbb{R}^d$.*

The **Definition** (5) presents an important concept for stochastic processes in high-dimensional cases, and this is a general description for the task to learn in mentioned related works. This includes but is not limited to GPs and characterizes the distribution over the stochastic function family.

A.2 *Proof of Proposition 1*

As we know, a Gaussian distribution is closed under marginalization, conditional probability computation, and some other trivial operations. Here the statistical parameter invariance towards the order of the context variables in per sample predictive distribution would be demonstrated.

Given a multivariate Gaussian as the context

$$X = [x_1, x_2, \ldots, x_N]^T \sim \mathcal{N}(X; [\mu_1, \mu_2, \ldots, \mu_N]^T, \Sigma(x_1, x_2, \ldots, x_N))$$

, and for any permutation operation over the order $\pi : [1, 2, \ldots, N] \to [\pi_1, \pi_2, \ldots, \pi_N]$, there exist a permutation matrix $P_\pi = [e_{\pi_1}^T, e_{\pi_2}^T, \ldots, e_{\pi_N}^T]$, where only the π_i-th position is one with the rest zeros. Naturally, it results in a permutation over the random variables in coordinates.

$$P_\pi [x_1, x_2, \ldots, x_N]^T = [x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_N}]^T = X_\pi \tag{8.6}$$

The random variable X_π follows another multivariate Gaussian as

$$X_\pi \sim \mathcal{N}(X_\pi; P_\pi \mu, P_\pi \Sigma P_\pi^T) = \mathcal{N}(X_\pi; \mu_\pi, \Sigma_\pi)$$

. In an elementwise way, we can rewrite the statistics in the form as follows.

$$\begin{aligned} &\mathbb{E}[x_{\pi_i}] = \mu_{\pi_i} \\ &\sigma_{ls}^\pi = e_l^T \Sigma_\pi e_s = e_l^T P_\pi \Sigma P_\pi^T e_s = e_{\pi_l} \Sigma e_{\pi_s} = cov(x_{\pi_l}, x_{\pi_s}) \end{aligned} \tag{8.7}$$

Notice in Eq. (8.7), the statistics are *permutation equivariant* now.

As the most critical component in GPs, the predictive distribution conditioned on the context $D = [X_{1:N}, Y_{1:N}]$ can be analytically computed once GPs are well trained and result in some mean function m_θ.

$$\begin{aligned} &p(y_*|Y_{1:N}) = \mathcal{N}(y_*|\tilde{\mu}, \tilde{\sigma}^2) \\ &\tilde{\mu} = m_\theta(x_*) + \Sigma_{x_*,D} \Sigma_{D,D}^{-1} (y_D - m_\theta(x_D)) \\ &\tilde{\sigma}^2 = \sigma_{x_*,x_*}^2 - \Sigma_{x_*,D} \Sigma_{D,D} \Sigma_{D,x_*} \end{aligned} \tag{8.8}$$

Similarly, after imposing a permutation π over the order of elements in the context, we can compute the first and second order of statistics between $D_\pi = [X_{\pi(1:N)}, Y_{\pi(1:N)}]$ and per target point $[x_*, y_*]$.

$$\begin{aligned} &\Sigma_{x_*,D_\pi} = \Sigma_{x_*,D} P_\pi^T = P_\pi \Sigma_{D,x_*} \\ &\Sigma_{D_\pi,D_\pi}^{-1} = P_\pi \Sigma_{D,D}^{-1} P_\pi^T \\ &y_{D_\pi} - m_\theta(x_{D_\pi}) = P_\pi (y_D - m_\theta(x_D)) \end{aligned} \tag{8.9}$$

Hence, with the property of orthogonality of permutation matrix P_π, it is easy to verify the *permutation invariance* in statistics for per target predictive distribution.

$$\begin{aligned} &\Sigma_{x_*,D} \Sigma_{D,D}^{-1} (y_D - m_\theta(x_D)) = \Sigma_{x_*,D_\pi} \Sigma_{D_\pi,D_\pi}^{-1} (y_{D_\pi} - m_\theta(x_{D_\pi})) \\ &\Sigma_{x_*,D} \Sigma_{D,D} \Sigma_{D,x_*} = \Sigma_{x_*,D_\pi} \Sigma_{D_\pi,D_\pi} \Sigma_{D_\pi,x_*} \end{aligned} \tag{8.10}$$

To inherit such a property, NP employs a permutation invariant function in embeddings, and the predictive distribution in NP models is invariant to the order of context points. Also, when multiple target samples exist in the predictive distribution, it is trivial that the statistics between the context and the target in a GP predictive distribution are *permutation equivariant* in terms of the order of target variables.

A.3 *Proof of DSVNP as Exchangeable Stochastic Process*

In the main passage, we formulate the generation of DSVNP as:

$$\rho_{x_{1:N+M}}(y_{1:N+M}) = \iint \prod_{i=1}^{N+M} p(y_i|z_G, z_i, x_i)p(z_i|x_i, z_G)p(z_G)dz_{1:N+M}dz_G \tag{8.11}$$

which indicates the scenario of any finite collection of random variables in $\boldsymbol{y}$-space. Our intention is to show this induces an exchangeable stochastic process. Equivalently, two conditions for Kolmogorov Extension Theorem are required to be satisfied.

- **Marginalization Consistency.** Generally, when the integral is finite, the swap of orders in integration is allowed. Without exception, Eq. **(8.11)** is assumed to be bounded with some appropriate distributions. Then, for the subset of indexes $\{N+1, N+2, \ldots, N+M\}$ in random variables $\boldsymbol{y}$, we have:

$$\begin{aligned}
\int \rho_{x_{1:N+M}}(y_{1:N+M})dy_{N+1:N+M} &= \iiint \prod_{i=1}^{N+M} p(y_i|z_G, z_i, x_i) \\
&\quad p(z_i|x_i, z_G)p(z_G)dz_{1:N+M}dz_G dy_{N+1:N+M} \\
= \iint \prod_{i=1}^{N} p(y_i|z_G, z_i, x_i)p(z_i|x_i, z_G)\Big[\iint \prod_{i=N+1}^{N+M} & p(y_i|z_G, z_i, x_i)p(z_i|x_i, z_G) \\
&\quad dy_{N+1:N+M}dz_{N+1:N+M}\Big]p(z_G)dz_G dz_{1:N} \\
= \iint \prod_{i=1}^{N} p(y_i|z_G, z_i, x_i)p(z_i|x_i, z_G)p(z_G)dz_{1:N}dz_G &= \rho_{x_{1:N}}(y_{1:N})
\end{aligned} \tag{8.12}$$

  hence, the marginalization consistency is verified.

- **Exchangeability Consistency.** For any permutation π towards the index set $\{1, 2, \ldots, N\}$, we have:

$$\begin{aligned}
\rho_{x_{1:N}}(y_{1:N}) &= \iint \prod_{i=1}^{N} p(y_i|z_G, z_i, x_i)p(z_i|x_i, z_G)p(z_G)dz_{1:N}dz_G \\
&= \iint \prod_{i=1}^{N} \Big[p(y_{\pi_i}|z_G, z_{\pi_i}, x_{\pi_i})p(z_{\pi_i}|x_{\pi_i}, z_G)dz_{\pi_i}\Big]p(z_G)dz_G \\
= \iint \prod_{i=1}^{N} p(y_{\pi_i}|z_G, z_{\pi_i}, x_{\pi_i})p(z_{\pi_i}|x_{\pi_i}, z_G)p(z_G)dz_{\pi_{(1:N)}}dz_G &= \rho_{x_{\pi(1:N)}}(y_{\pi(1:N)})
\end{aligned} \tag{8.13}$$

  hence, the exchangeability consistency is demonstrated as well.

With properties in Eq. (8.12) and (8.13) verified, our proposed DSVNP is an exchangeable stochastic process in this case.

A.4 *Derivation of Evidence Lower Bound for DSVNP*

Akin to vanilla NPs, we assume the existence of a global latent variable z_G, which captures summary statistics consistent between the context $[x_C, y_C]$ and the complete target $[x_T, y_T]$. With the involvement of an approximate distribution $q\big(z_G|[x_C, y_C, x_T, y_T]\big)$, we can naturally have an initial ELBO in the following form.

$$\begin{aligned}
&\ln\Big[p(y_*|x_C, y_C, x_*)\Big] = \ln\Big[\mathbb{E}_{q(z_G|x_C,y_C,x_T,y_T)} p(y_*|z_G, x_*)\frac{p(z_G|x_C, y_C)}{q(z_G|x_C, y_C, x_T, y_T)}\Big] \\
&\geq \mathbb{E}_{q(z_G|x_C,y_C,x_T,y_T)} \ln\Big[p(y_*|z_G, x_*)\Big] - D_{KL}\Big[q(z_G|x_C, y_C, x_T, y_T) \parallel p(z_G|x_C, y_C)\Big]
\end{aligned} \tag{8.14}$$

Note that in Eq. (8.14), the conditional prior distribution $p(z_G|x_C, y_C)$ is intractable in practice, and the approximation is used here, and such a prior is employed to infer the global latent variable in testing processes. For the approximate posterior $q(z_G|x_C, y_C, x_T, y_T)$, it uses the context and the full target information, and the sample $[x_*, y_*]$ is just an instance in the full target.

Further, by introducing a target specific local latent variable z_*, we can derive another ELBO for the prediction term on the right side of Eq. (8.14) with the same trick.

$$\begin{aligned}
&\mathbb{E}_{q(z_G|x_C,y_C,x_T,y_T)} \ln\Big[p(y_*|z_G, x_*)\Big] \\
&= \mathbb{E}_{q(z_G|x_C,y_C,z_T,y_T)} \ln\Big[\mathbb{E}_{q(z_*|z_G,[x_*,y_*])} p(y_*|z_G, z_*, x_*)\frac{p(z_*|z_G, x_*)}{q(z_*|z_G, [x_*, y_*])}\Big] \\
&\geq \mathbb{E}_{q(z_G|x_C,y_C,x_T,y_T)}\mathbb{E}_{q(z_*|z_G,[x_*,y_*])} \ln\Big[p(y_*|z_G, z_*, x_*)\Big] \\
&-\mathbb{E}_{q(z_G|x_C,y_C,x_T,y_T)}\Big[D_{KL}[q(z_*|z_G, [x_*, y_*]) \parallel p(z_*|z_G, x_*)]\Big]
\end{aligned} \tag{8.15}$$

With the combination of Eq. (8.14) and (8.15), the final ELBO $\mathcal{L}$ as the right term in the following is formulated.

$$\begin{aligned}
\ln\Big[\underbrace{p(y_*|x_C, y_C, x_*)}_{\textit{implicit data likelihood}}\Big] \geq\ & \mathbb{E}_{q_{\phi_1}(z_G)}\mathbb{E}_{q_{\phi_2}(z_*)} \ln[\underbrace{p(y_*|z_G, z_*, x_*)}_{\textit{data likelihood}}] \\
&-\mathbb{E}_{q_{\phi_1}(z_G)}[D_{KL}[q(z_*|z_G, x_*, y_*) \parallel \underbrace{p(z_*|z_G, x_*)}_{\textit{local prior}}]\Big] \\
&-D_{KL}\Big[q(z_G|x_C, y_C, x_T, y_T) \parallel \underbrace{p(z_G|x_C, y_C)}_{\textit{global prior}}\Big]
\end{aligned} \tag{8.16}$$

The real data likelihood is generally implicit, and the ELBO is an approximate objective. Note that the conditional prior distribution in Eq. (8.16), $p(z_*|z_G, x_*)$ functions as a local latent variable and is approximated with a Gaussian distribution for the sake of easy implementation. With reparameterization trick, used as: $z_G = \mu_{\phi_1} + \epsilon_1\sigma_{\phi_1}$ and

Table 12: Pointwise Average Negative Log-likelihoods for 2000 realizations. Rows with J consider all data points, including the context, while those with P exclude the context points in statistics. (Figures in brackets are variances.)

Prediction	CNP	NP	AttnNP	DSVNP
Inter(J)	NaN	-0.958±2E-5	**-1.149±8E-6**	-0.975±2E-5
Inter(P)	-0.802±1E-6	-0.949±2E-5	**-1.141±6E-6**	-0.970±2E-5
Extra(J)	NaN	8.192±7E1	8.091±7E2	**4.203±9E0**
Extra(P)	**1.764±1E-1**	8.573±8E1	8.172±7E2	4.303±1E1

$z_* = \mu_{\phi_2} + \epsilon_2\sigma_{\phi_2}$, we can estimate the gradient towards the sample (x_*, y_*) analytically in Eq. (8.17), (8.18) and (8.19).

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\phi_1} &= \mathbb{E}_{\epsilon_1\sim N(0,I)}\mathbb{E}_{\epsilon_2\sim N(0,I)}\frac{\partial}{\partial\phi_1}\ln\Big[p(y_*|\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1},\mu_{\phi_2}+\epsilon_2\sigma_{\phi_2},x_*)\Big]\\ -\mathbb{E}_{\epsilon_1\sim N(0,I)}\frac{\partial}{\partial\phi_1}&D_{KL}\Big[q(\mu_{\phi_2}+\epsilon_2\sigma_{\phi_2}|\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1},x_*,y_*)\parallel p(\mu_{\phi_2}+\epsilon_2\sigma_{\phi_2}|\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1},x_*)\Big]\\ -\mathbb{E}_{\epsilon_1\sim N(0,I)}\frac{\partial}{\partial\phi_1}&D_{KL}\Big[q(\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1}|x_C,y_C,x_T,y_T)\parallel p(\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1}|x_C,y_C)\Big]\end{aligned} \tag{8.17}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\phi_2} &= \mathbb{E}_{\epsilon_1\sim N(0,I)}\mathbb{E}_{\epsilon_2\sim N(0,I)}\frac{\partial}{\partial\phi_2}\ln\Big[p(y_*|\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1},\mu_{\phi_2}+\epsilon_2\sigma_{\phi_2},x_*)\Big]\\ -\mathbb{E}_{\epsilon_1\sim N(0,I)}\frac{\partial}{\partial\phi_2}&D_{KL}\Big[q(\mu_{\phi_2}+\epsilon_2\sigma_{\phi_2}|\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1},x_*,y_*)\parallel p(\mu_{\phi_2}+\epsilon_2\sigma_{\phi_2}|\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1},x_*)\Big]\end{aligned} \tag{8.18}$$

$$\frac{\partial\mathcal{L}}{\partial\theta} = \mathbb{E}_{\epsilon_1\sim N(0,I)}\mathbb{E}_{\epsilon_2\sim N(0,I)}\frac{\partial}{\partial\theta}\ln\Big[p_\theta(y_*|\mu_{\phi_1}+\epsilon_1\sigma_{\phi_1},\mu_{\phi_2}+\epsilon_2\sigma_{\phi_2},x_*)\Big] \tag{8.19}$$

A.5 *Implementation Details in Experiments*

Unless explicitly mentioned, otherwise we make of an one-step amortized transformation as $dim\_lat \mapsto [\mu\_lat, \ln\sigma\_lat]$ to approximate parameters of the posterior in NP models. Especially for DSVNP, the approximate posterior of a local latent variable is learned with the neural network transformation in the approximate posterior $[dim\_lat, dim\_latx, dim\_laty] \mapsto dim\_lat$ and the prior network $[dim\_lat, dim\_latx] \mapsto dim\_lat$. (For the sake of simplicity, these are not further mentioned in tables of neural structures.) All models are trained with Adam and implemented on Pytorch.

Table 13: Neural Network Structure of (C)NP Models for 1-D Stochastic Process. The transformations in the table are in linear form, followed by ReLU activation mostly. And Dropout rate for DNN is defined as 0.5 for all transformation layers in Encoder. As for AttnNP and DSVNP, the encoder network is doubled in the table since some local variable exists for prediction.

NP Models	Encoder	Decoder
CNP&NP	$[dim\_x, dim\_y] \mapsto 32 \mapsto 32 \mapsto dim\_lat$	$[dim\_x, dim\_lat] \mapsto 2 * dim\_y$
AttnNP&DSVNP	$[dim\_x, dim\_y] \mapsto 32 \mapsto 32 \mapsto dim\_lat$	$[dim\_x, 2 * dim\_lat] \mapsto 2 * dim\_y$

Synthetic Experiments

For synthetic experiments, all implementations resemble that in Attentive NPs [1]. Moreover, the neural structures for NPs are reported in Table (13), where dim_lat is 128. Note that for the amortized transformations in encoders of NP, AttnNP and DSVNP, we use the network to learn the distribution parameters as: $dim\_lat \mapsto [\mu_{lat}, \ln \sigma_{lat}]$. In the training process, the maximum number of iterations for all (C)NPs is 800k,, and the learning rate is 5e-4. For the testing process, the maximum number of context points in interpolation tasks is 50, while that in extrapolation tasks is 200. Note that the coefficient for KL divergence terms in (C)NPs is set 1 as default. However, for DSVNP, we assign more penalty to KL divergence term of a local latent variable to avoid overfitting, where the weight is set $\beta_2 = 1000$ for simplicity. Admittedly, more penalty to such a term reduces prediction accuracy, and some dynamically tuning such parameter would bring some promotion in accuracy.

System Identification Experiments

In the Cart-Pole simulator, the input of the system is the vector of the coordinate/angle and their first order derivative and a random action $[x_c, \theta, x_c', \theta', a]$, while the output is the transited state in the next time step $[x_c, \theta, x_c', \theta']$. The force as the action space ranges between [-10,10] N, where the action is a randomly selected force value to impose in the system. For the training dataset from 6 configurations of environments, we sample 100 state transition pairs for each configuration as the maximum context points. These context points work as the identification of a specific configuration. The neural architectures for CNP, NP, AttnNP and DSVNP refer to Table (14), and default parameters are listed in $\{dim\_latxy = 32, dim\_lat = 32, dim\_h = 400\}$. All neural network models are trained with the same learning rate 1e-3. The batch size and the maximum number of epochs are 100. For AttnNP, we notice the generalization capability degrades with the training process, so early stopping is used. For DSVNP, the regularization weight is set as $\{\beta_1 = 1, \beta_2 = 5\}$, while the KL divergence term weight is fixed as 1 for NP and AttnNP.

1 https://github.com/deepmind/neural-processes

Table 14: Neural Network Structure of (C)NP Models in System Identification Tasks. The transformations in the table are linear, followed with ReLU activation mostly. As for AttnNP and DSVNP, the encoder network is doubled in the table since some local variable exists for prediction. Here only dot product attention is used in AttnNP.

NP Models	Encoder	Decoder
CNP&NP	$[dim\_x, dim\_y] \mapsto \underbrace{dim\_latxy \mapsto dim\_latxy}_{2\ times}$ $dim\_latxy \mapsto dim\_lat.$	$[dim\_x, dim\_lat] \mapsto dim\_h \mapsto dim\_h$ $dim\_h \mapsto 2 * dim\_y$
AttnNP&DSVNP	$[dim\_x, dim\_y] \mapsto \underbrace{dim\_latxy \mapsto dim\_latxy}_{2\ times}$ $dim\_x \mapsto dim\_latx$; $[dim\_latx, dim\_laty] \mapsto dim\_lat.$	$[dim\_x, 2 * dim\_lat] \mapsto dim\_h \mapsto dim\_h$ $dim\_h \mapsto 2 * dim\_y$

Multi-output Regression Experiments

SARCOS records inverse dynamics for an anthropomorphic robot arm with seven degree freedoms, and the mission is to predict 7 joint torques with 21-dimensional state space (7 joint positions, 7 joint velocities and 7 accelerations). WQ targets to predict the relative representation of plant and animal species in Slovenian rivers with some physical and chemical water quality parameters. SCM20D is some supply chain time series dataset for many products, while SCFP records online click logs for several social issues.

Before data split, standardization over input and output space is operated on a dataset, scaling each dimension of the dataset in zero mean and unit variance [2]. Such pre-processing is required to ensure the stability of training. Also, we find that directly treating the data likelihood term as some Gaussian and optimizing the negative log-likelihood of Gaussian to learn both mean and variance do harm to the prediction. Hence average MSE is selected as the objective. As for the variance estimation for uncertainty, Monte Carlo estimation can be used. For all dataset, we employ the neural structure in Table (15), and default parameters in Encoder and Decoder are in the list $\{dim\_h = 100, dim\_latx = 32, dim\_laty = 8, dim\_lat = 64\}$. The learning rate for Adam is selected as 1e-3, the batch size for all datasets is 100, the maximum number of context points is randomly selected during training, and the maximum epochs in training are up to the scale of the dataset and convergence condition. Here the maximum epochs are respectively 300 for SARCOS, 3000 for SCM20D, and 5000 for WQ. For the testing process, 30 data points are randomly selected for each batch's prediction context. Also, the hyper-parameters as the weights of KL divergence term are the same in implementation as one without additional modification in this experiment.

2 https://scikit-learn.org/stable/modules/preprocessing.html

Table 15: Neural Network Structure of (C)NP Models in Multi-Output Regression Tasks. The transformations in the table are linear, followed with ReLU activation mostly. And Dropout rate for DNN is defined as 0.01 for all transformation layers in Encoder. As for AttnNP and DSVNP, the encoder network is doubled in the table since there exist some local variable for prediction. Here only dot product attention is used in AttnNP.

NP Models	Encoder	Decoder
DNN(MC-Dropout)	$dim\_x \mapsto \underbrace{dim\_h \mapsto dim\_h}_{2\ times}$	$dim\_lat \mapsto dim\_h$
	$dim\_h \mapsto dim\_lat$	$dim\_h \mapsto dim\_y$
CNP&NP	$dim\_x \mapsto \underbrace{dim\_h \mapsto dim\_h}_{2\ times} \mapsto dim\_latx$	$[dim\_latx, dim\_lat] \mapsto dim\_h$
	$dim\_y \mapsto dim\_laty$; $[dim\_latx, dim\_laty] \mapsto dim\_lat.$	$dim\_lat \mapsto dim\_y$
AttnNP&DSVNP	$dim\_x \mapsto \underbrace{dim\_h \mapsto dim\_h}_{2\ times} \mapsto dim\_latx$	$[dim\_latx, 2 * dim\_lat] \mapsto dim\_h$
	$dim\_y \mapsto dim\_laty$; $[dim\_latx, dim\_laty] \mapsto dim\_lat.$	$dim\_lat \mapsto dim\_y$

Image Classification and O.O.D. Detection

The implementations of NP related models and the Monte-Carlo Neural Network are pretty similar. On MNIST task, the feature extractor for images is taken as LeNet-like structure as [20, 'M', 50, 'F', '500'][3], and the decoder is one-layer transformation. On CIFAR10 task, the extractor is parameterized in VGG-style network as [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, 512, 512, 'M', 512, 512, 512, 512, 'M'][4], and the decoder is also one-layer transformation from latent variable to label output in softmax form. Other parameters are in the list $\{dim\_latx = 32, dim\_laty = 64, dim\_lat = 64\}$ for MNIST and $\{dim\_latx = 32, dim\_laty = 64, dim\_lat = 128\}$. Both labels are represented in a one-hot encoding way and then further transformed to some continuous embedding. Batch size in training is 100 as default, the number of context samples for NP related models is randomly selected no larger than 100 in each batch, while the optimizer Adam is with learning rate $1e^{-3}$ for MNIST task and $5e^{-5}$ for CIFAR10 task. The maximum epochs for both are 100 in both cases, and the size of all sources and o.o.d. dataset is 10000. Dropout rates for MC-Dropout in encoder networks are respectively 0.1 and 0.2 for LeNet-like ones and VGG-like ones. In the testing process, 100 samples from the source dataset are randomly selected as the context points.

Before the prediction process (estimating predictive entropies on both domain dataset and o.o.d dataset), images on MNIST are normalized in the interval $[0, 1]$, those on CIFAR10 are standardized as well, and all o.o.d. datasets follow a similar way as that on MNIST or CIFAR10. More specifically, the Rademacher dataset is generated in the way:

3 Numbers are dimensions of Out-Channel with kernel size 5, 'F' is flattening operation, and each layer is followed with ReLU activation.

4 Numbers are dimensions of Out-Channel with kernel size 3 and padding 1 in each layer, followed with BatchNorm and ReLU function, here M means max-polling operation.

Table 16: Neural Network Structure of (C)NP Models in Image Classification Tasks. The transformations in the table are linear, followed by ReLU activation mostly. And Dropout rate for DNN is defined as 0.5 for all transformation layers in Encoder. As for AttnNP and DSVNP, the encoder network is doubled in the table since some local variables exist for prediction.

NP Models	Encoder	Decoder
DNN(MC-Dropout)	$\underbrace{dim\_x \mapsto dim\_h}_{embedding\ net}$ $dim\_h \mapsto dim\_lat$	$dim\_lat \mapsto dim\_y$
CNP&NP	$\underbrace{dim\_x \mapsto dim\_h}_{embedding\ net} \mapsto dim\_latx$ $dim\_y \mapsto dim\_laty$; $[dim\_latx, dim\_laty] \mapsto dim\_lat$.	$[dim\_latx, dim\_lat] \mapsto dim\_y$
AttnNP&DSVNP	$\underbrace{dim\_x \mapsto dim\_h}_{embedding\ net} \mapsto dim\_latx$ $dim\_y \mapsto dim\_laty$; $[dim\_latx, dim\_laty] \mapsto dim\_lat$.	$[dim\_latx, 2 * dim\_lat] \mapsto dim\_y$

place bi-nominal distribution with probability 0.5 over in image-shaped tensor and then minus 0.5 to ensure the zero-mean in statistics. A similar operation is taken in uniform cases, while the Gaussian o.o.d. dataset is from standard Normal distribution. All results are reported in Table (6).

B SUPPLEMENTARY MATERIALS IN CHAPTER 4

B.1 *Frequently Asked Questions*

Here we collected some frequently asked questions from reviewers and other literature researchers. We thank these reviewers for these precious questions and add more explanations.

Selection of Benchmarks. Admittedly, NP variants can be applied to a series of downstream tasks. Our selection of benchmark missions is based on existing literature for NP models. The system identification task was previously investigated with NP variants in work (Galashov et al., 2019; Volpp et al., 2020), in which learning transferable physics dynamics with NPs is a crucial application. The image completion task is more commonly used in work (Zintgraf et al., 2019; Garnelo et al., 2018a,b). The meta reinforcement learning task can also be studied within NP framework (Gondal et al., 2021; Eslami et al., 2018).

NP Family in Classification Tasks. We have tried to search neural architectures for few-shot image classification tasks, but the performance is not ideal compared to other metrics based methods. Meanwhile, we have gone through most NP related work, and it is challenging to achieve SOTA few-shot image classification results with standard neural architectures in NPs, *e.g.* multi-layer perceptrons (MLPs). Maybe this is due to the nature of stochastic processes, which can address regression problems more efficiently. Unless specialized modules instead of MLPs are used, we do not expect NP variants with MLPs can achieve SOTA performance. The aim of this paper focuses more on a mixture of expert inductive biases and places less attention on neural architecture search. So MLPs are shared across all baselines to enable a fair comparison.

Expressiveness of Mixture of Expert Inductive Biases. A natural question about learning diverse functional representation is whether these multiple expert latent variables will collapse into one. We refer the collapsed representation to vanilla (C)NPs, and the previous empirical results show the collapsed ones work poorer than the mixture of expert ones in both few-shot regression tasks and meta reinforcement learning tasks. Also, from the multimodal simulation result, we discover that both latent assignment variables and expert variables are meaningful, which reflects the effectiveness of mixture expert inductive biases. That means we have not encountered meta representation collapse issues in experiments.

Extension with other Mixture of Experts Models. Our work is the first to examine MoEs inductive biases in NPs family, and the used MoE module is an amortized inference one. We have not found a trivial implementation of MoEs in the meta learning domain. Nevertheless, in MoEs literature, other more effective MoE models exist, which can better trade-off communication/memory and performance. So NPs families can also be combined with these models, such as GShard (Lepikhin et al., 2020), Deepsepeed MoEs (Rajbhandari et al., 2022) and etc.

Potential Applications in Industry. Here we provide two available applications with MoE-NPs in the industry. One is in multilingual machine translation or multilingual language auto-completion. In this case, a mixture of experts corresponds to multilingual functional priors for multi-modal signals (Shi et al., 2019) and enables the prediction with partial observations. Another application lies in modelling irregular time series (Kidger et al., 2020; Schirmer et al., 2022). In this case, diverse experts can handle discontinuous components in a wealthy family of stochastic functions. Meanwhile, the entropy of learned assignment latent variables can tell us the regions likely to be discontinuous, which is quite helpful in anomaly detection in a black-box system.

B.2 *Probabilistic Graphs in Meta Training/Testing*

As exhibited in Fig. (25)/(26), shown are computational diagrams when implementing MoE-NPs in meta learning tasks.

Variational Posteriors. Since the real posteriors for both $\{z_k\}_{k=1}^K$ and e_T are computationally intractable, the approximate ones are used in practice. These are called variational posteriors, e.g. $q_{\phi_{1,k}}$ for an expert latent variable z_k and $q_{\phi_{2,1}}(e|x, y, z_{1:K})$ for assignment latent variables e. For the sake of simplicity, we denote $\{q_{\phi_{1,k}}\}_{k=1}^K$ by q_{ϕ_1} in some time. Importantly, the Gumbel-softmax trick (Jang et al., 2016) is used to sample assignment latent variable e from categorical approximate distributions.

Variational Priors. In some cases, the prior distributions for $\{z_k\}_{k=1}^K$ and e_T are set to constrain the scope of prior distributions. For example, in few-shot supervised learning, since the context and the target have the same form, the variational prior is selected to be q_{ϕ_1} as well to ensure consistency, and this works in meta-testing phases. For the assignment latent variable e, this uses the same form in conditional VAE (Sohn et al., 2015) as $p_{\phi_{2,2}}$.

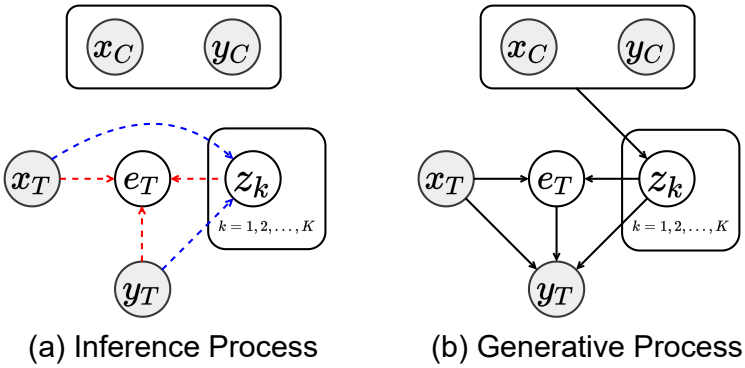


Figure 25: Computational Diagram in Few-shot Supervised Learning. Blue dotted lines are for expert latent variables, while red dotted lines are for assignment latent variables in inference.

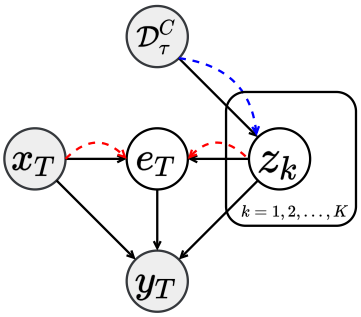


Figure 26: Computational Diagram in Meta Reinforcement Learning. This is in an information bottleneck form (Rakelly et al., 2019; Gondal et al., 2021). The variational posteriors are $p_{\phi,k} = q_{\phi_{1,k}}(z|\mathcal{D}_\tau^C)$ and $q_{\phi_2} = q_{\phi_{1,k}}(e|s, z_{1:K})$. As for the variational prior, we use the same strategy as that in (Rakelly et al., 2019). They are respectively the fixed normal $p(z) = \mathcal{N}(0, I)$ and the categorical $p(e) = \texttt{Cat}(e; K, [\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}])$.

B.3 *More Descriptions of NP Family Models and Meta RL*

In the main paper, we unify the description of NP family models in both few-shot supervised learning and meta reinforcement learning. This is the same as that in FCRL (Gondal et al., 2021). Meta learning in NP related models is to learn functional representations of different tasks and formulate the fast adaptation via inferring the task specific conditional distribution $p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C) = \int p(\mathcal{D}_\tau^T|z)p(z|\mathcal{D}_\tau^C)dz$ (equivalent to Eq. (4.1)).

To clarify the downstream reinforcement learning task using NP family models, we add the following explanations. In few-shot supervised learning, $\mathcal{D}_\tau^C$ and $\mathcal{D}_\tau^T$ are of the same form. However, in context-based meta reinforcement learning, $\mathcal{D}_\tau^C$ is a set of task-specific transitions, and $\mathcal{D}_\tau^T$ is a set of state (action) values. As a result, the approximate posteriors and the selected priors to resolve Eq. (4.1) are distinguished in separate meta learning cases.

In context-based meta reinforcement learning, we can translate our problem into finding the distribution of optimal value functions in Eq. (4.5); this corresponds to learning meta critic modules with NP family models. Given a transition sample $[s, a, r(s, a), s']$, the target input is the state $x_T = s$ and the target output is the temporal difference target $y_T = \hat{Q}(s, a) = r(s, a) + \gamma V([s', z'])$. The standard Gaussian distribution is used as the prior $p(z_k) = \mathcal{N}(0, I)$ in Eq. (4.14), while the approximate posterior is learned from $\mathcal{D}_\tau^C$ with permutation invariant functions.

In total, sampling from the state-dependent approximate posterior $z \sim q_\phi(z|s, \mathcal{D}_\tau^C)$ corresponds to Eq. (8.20), where the operator $\odot$ denotes the selection process with help of Hadamard products.

$$z_{1:K} \sim q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^C),\ e \sim q_{\phi_2}(e|s, z_{1:K}),\ z = z_{1:K} \odot e \tag{8.20}$$

B.4 *MoE-NPs as Exchangeable $\mathcal{SP}$s*

Generative Processes

To better understand our developed model in meta learning scenarios, we translate Eq. (4.6) into a step-wise generative process. The same with that in the main paper, the task distribution is denoted by $p(\mathcal{T})$, and we presume K-experts to summarize the stochastic traits of a task.

$$\tau \sim p(\mathcal{T}), \quad z_k \sim p(z_k|\mathcal{T}) \; \forall k \in \{1, 2, \ldots, K\} \tag{8.21a}$$

$$x \sim p(x), \quad e \sim \prod_{k=1}^{K} \alpha_k(x, z_{1:K})^{\mathbb{I}[e_k=1]}, \quad z = [z_1, z_2, \ldots, z_K]^T \odot e \tag{8.21b}$$

$$[\mu_x, \Sigma_x] = g_\theta(x, z), \quad y \sim \mathcal{N}(\mu_x, \Sigma_x + \epsilon^2 I) \tag{8.21c}$$

Here a MoE-NP for the task τ is specified with K-expert latent variables $z_{1:K}$ in Eq. (8.21.a). The probability mass function for a data point related categorical distribution $\texttt{Cat}(K, \alpha(x, z_{1:K}))$ is denoted by $p(e|x, z_{1:K}) = \prod_{k=1}^{K} \alpha_k(x, z_{1:K})^{\mathbb{I}[e_k=1]}$ in Eq. (8.21.b), and e is an assignment latent variable to select an expert for the generative process. After that, the distributional parameters for the output of a data point are learned via a function g_θ in Eq. (8.21.c), followed by the output distribution $\mathcal{N}(\mu_x, \Sigma_x + \epsilon^2 I)$.

Note that a collection of sampled functional experts are represented in a vector of variables $z_{1:K} = [z_1, z_2, \ldots, z_K]^T$ and $e = [0, \cdots, \underbrace{1}_{k\text{-th position}}, \cdots, 0]^T \Leftrightarrow e_k = 1$ is a one-hot vector in Eq. (8.21.b). In Eq. (8.21.c), the expert is selected in a way $z = z_{1:K} \odot e$. For the sake of generality, irreducible noise $\mathcal{N}(0, \epsilon^2 I)$ is injected into the output. In experiments, K-expert latent variables $z_{1:K}$, as well as discrete assignment latent variables e, are non-observable.

Consistency Properties

Remember that the generative model is induced in the main paper as follows. And we claim that *our designed generative model MoE-NP formulates a family of exchangeable $\mathcal{SP}$* in **Definition** (5).

$$\rho_{x_{1:N}}(y_{1:N}) = \int \prod_{k=1}^{K} p(z_k) \prod_{i=1}^{N} \left[\sum_{k=1}^{K} p(y_i|x_i, z_{1:K}, e_k = 1) p(e_k = 1|x_i, z_{1:K}) \right] dz_{1:K} \tag{8.22}$$

So it is necessary to verify two formerly mentioned consistencies according to Kolmogorov Extension Theorem (Bhattacharya and Waymire, 2009) and de Finetti's Theorem (De Finetti, 1937). This is to show the existence of $\mathcal{SP}$s in Eq. (8.22).

Exchangeability Consistency. For N data points from Eq. (8.22), we impose any permutation operation σ over their indices, and this results in $\sigma : [1, 2, \ldots, N] \rightarrow [\sigma_1, \sigma_2, \ldots, \sigma_N]$. Then we can check that the following equation is satisfied since the element-wise product of probabilities can be swapped.

$$\begin{aligned} \rho_{x_{1:N}}(y_{1:N}) &= \int \left[\prod_{k=1}^{K} p(z_k)\right] dz_{1:K} \\ &\prod_{i=1}^{N}\left[\sum_{k=1}^{K} p(y_{\sigma_i}|x_{\sigma_i}, z_{1:K}, e_k = 1) p(e_k = 1|x_{\sigma_i}, z_{1:K})\right] \\ = \int \left[\prod_{k=1}^{K} p(z_k)\right] &\prod_{i=1}^{N}\left[\sum_{k=1}^{K} p(y_{\sigma_i}|x_{\sigma_i}, z_{1:K}, e_k = 1) p(e_k = 1|x_{\sigma_i}, z_{1:K})\right] dz_{1:K} \\ &= \rho_{x_{\sigma(1:N)}}(y_{\sigma(1:N)}) \quad \square \end{aligned} \tag{8.23}$$

Marginalization Consistency. Given the assumption that the integral in Eq. (8.22) is finite, we pick up a subset of indices $[M+1, M+2, \ldots, N]$ and make $M < N$ without difference in orders. Furthermore, the result after marginalization over y-variable in the selected indices can be verified based on the following equation.

$$\begin{aligned} \int \rho_{x_{1:N}}(y_{1:N}) dy_{M+1:N} &= \int \left[\prod_{k=1}^{K} p(z_k)\right] dz_{1:K} \left[\int \prod_{i=1}^{N} p(y_i|x_i, z_{1:K}) dy_{M+1:N}\right] \\ &= \iint \prod_{i=1}^{M} p(y_i|x_i, z_{1:K}) \prod_{i=M+1}^{N} (p(y_i|x_i, z_{1:K}) \prod_{k=1}^{K} p(z_k) dz_{1:K} dy_{M+1:N} \\ &= \int \left[\prod_{k=1}^{K} p(z_k)\right] dz_{1:K} \prod_{i=1}^{M}\left[\sum_{k=1}^{K} p(y_i|x_i, z_{1:K}, e_k = 1) p(e_k = 1|x_i, z_{1:K})\right] \\ &= \rho_{x_{1:M}}(y_{1:M}) \quad \square \end{aligned} \tag{8.24}$$

Built on these two sufficient conditions, our developed MoE-NP is a well-defined exchangeable $\mathcal{SP}$.

B.5 *Summary of Existing NP Related Models*

Comparison in Technical Details

Here we give a summary of the difference between MoE-NPs and existing typical Neural Process models in Table (23). Some crucial traits include forms of encoders and decoders structures, types of latent variables, and inductive biases injected in modeling. Especially, the inductive bias for MoE-NP is reduced to be multiple functional priors, which means a collection of expert neural processes to induce the generated dataset. A general inductive bias behind NPs related models is the modeling of exchangeable stochastic processes with cheap computations. This corresponds to a distribution of functions, termed as

`functional` in the Table. Note that the recognition model of NP models in Meta Training Scenarios is replaced with $q_\phi(z|[x_T, y_T])$ since all target points can be available, but in Meta Testing Scenarios, only $[x_C, y_C]$ are accessible.

Time Complexity

As for running time complexity, the vanilla NPs and CNPs are with $O(N+M)$, while MoE-NPs are with $O(K*(N+M))$ (making M predictions with N observations). In practice, the number of experts is small, so the increase in running time complexity can be ignored in practice. In contrast, traditional Gaussian processes are $O((N+M)^3)$ in terms of running time complexity.

Table 17: Summary of Typical Neural Process Related Models (Meta-Testing Scenarios). The recognition model and the generative model respectively correspond to the encoder and the decoder in the family of neural processes.

Models	Recognition Model	Generative Model	Inductive Bias
CNP (Garnelo et al., 2018a)	$z = f_\phi(x_C, y_C)$	$p_\theta(y\|[x,z])$	conditional functional
NP (Garnelo et al., 2018b)	$q_\phi(z\|[x_C, y_C])$	$p_\theta(y\|[x,z])$	global functional
AttnNP (Kim et al., 2019a, 2021)	$q_{\phi_1}(z\|[x_C, y_C])$ $f_{\phi_2}(z_*\|[x_C, y_C], x_*)$	$p_\theta(y\|[x,z,z_*])$	global functional local embedding
FCRL (Gondal et al., 2021)	$f_\phi(z\|[x_C, y_C])$	$p_\theta(y\|[x,z])$	contrastive functional
ConvNP (Foong et al., 2020)	$p_\phi(z\|[x_C, y_C])$	$p_\theta(y\|[x,z])$	convolutional functional
Conv-CNP (Gordon et al., 2020a)	$f_\phi(z_*\|[x_C, y_C], x_*)$	$p_\theta(y\|[x,z_*])$	convolutional functional
MoE-NP (Ours)	$q_{\phi_1}(z_{1:K}\|[x_C, y_C])$ $q_{\phi_{2,1}}(e\|z_{1:K}, x, y)$	$p_\theta(y\|[x,z_{1:K},e])$ $p_{\phi_{2,2}}(e\|z_{1:K}, x)$	multiple functional

Additional Literature Review

We include other related works in this subsection. In unsupervised learning, the Neural Statistician Model (Edwards and Storkey, 2017) is introduced to compute summary statistics inside the dataset. The Generative Query Network (Eslami et al., 2018), a variant of NPs for the visual sensory dataset, uses a latent variable to abstract scenes in high dimensions. To capture heteroscedastic noise inside the stochastic process, DSVNP (Wang and Van Hoof, 2020) induces latent variables at different levels. The functional neural processes infer the directed acyclic graph in the latent space and formulate flexible exchangeable stochastic processes for single task problems (Louizos et al., 2019). Inspired by self-supervised learning, (Gondal et al., 2021; Mathieu et al., 2021) propose to augment the neural process with contrastive losses. (Yoon et al., 2020) combines context and recurrent memories to formulate sequential neural processes (SNPs). Though there exist several NP variants, none of them consider injecting multiple functional inductive biases in modeling.

B.6 *Formulation of Evidence Lower Bounds*

Since functional priors reflected in the K-expert latent variables $z_{1:K}$ are learned via approximate distributions, this can be directly optimized within the variational inference framework. So we leave these out in this discussion. The difficulty of optimization principally comes from the involvement of discrete latent variables. We, therefore, discuss the chance of using another traditional optimization algorithm, called Expectation Maximization (EM) (Moon, 1996), in our settings. Omitting the K-expert latent variables $z_{1:K}$ and corresponding variational distributions, we take a closer look at the assignment latent variable e in the logarithm likelihood as $\ln\left(\sum_{k=1}^{K} p(y|x, z_{1:K}, e_k = 1)p(e_k = 1|x, z_{1:K})\right)$ and derive the corresponding EM algorithm.

Expectation(E)-Step: Note that the assignment variable e is discrete with the categorical probability function $p(e|x, z_{1:K}) = \texttt{Cat}(e; K, \alpha(x, z_{1:K}))$. This step is to update the posterior of the proportional coefficients $\alpha(x, z_{1:K})$ based on the last time step model parameters $\theta^{(t)}$.

$$\alpha_k^{(t+1)} = p(e_k = 1|x, z_{1:K}, y) \propto p(e_k = 1)p_{\theta^{(t)}}(y|x, z_{1:K}, e_k = 1) \tag{8.25}$$

Here the prior distribution $p(e)$ can be a commonly used one $\texttt{Cat}(K, [\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}])$ or the last time updated one $p^{(t)}(e)$. As a result, updated categorical distribution parameters are:

$$\alpha^{(t+1)} = \begin{bmatrix} \alpha_1^{(t+1)} \\ \alpha_2^{(t+1)} \\ \vdots \\ \alpha_K^{(t+1)} \end{bmatrix} = \begin{bmatrix} \frac{\exp\left((\ln\left(p_{\theta^{(t)}}(y|x,z_{1:K},e_1=1)\right))/\tau\right)}{\sum_{k=1}^{K}\exp\left((\ln\left(p_{\theta^{(t)}}(y|x,z_{1:K},e_k=1)\right))/\tau\right)} \\ \frac{\exp\left((\ln\left(p_{\theta^{(t)}}(y|x,z_{1:K},e_2=1)\right))/\tau\right)}{\sum_{k=1}^{K}\exp\left((\ln\left(p_{\theta^{(t)}}(y|x,z_{1:K},e_k=1)\right))/\tau\right)} \\ \vdots \\ \frac{\exp\left((\ln\left(p_{\theta^{(t)}}(y|x,z_{1:K},e_K=1)\right))/\tau\right)}{\sum_{k=1}^{K}\exp\left((\ln\left(p_{\theta^{(t)}}(y|x,z_{1:K},e_k=1)\right))/\tau\right)} \end{bmatrix} \tag{8.26}$$

where τ is the temperature parameter.

Maximization(M)-Step: Once the distributional parameter of assignment latent variables are updated, the next step is to maximize the logarithm likelihood as $\theta^{(t+1)} = \arg\max_\theta \sum_{(x,y)\in\mathcal{D}} \ln\left[p_{\theta^{(t)}}(y|x, z_{1:K}, e)\right]$ given the last time updated model parameter $\theta^{(t)}$. With the help of gradient ascent, this can be written as follows,

$$\begin{gathered} \theta^{(t+1)} \leftarrow \theta^{(t)} + \lambda \sum_{(x,y)\in\mathcal{D}} \nabla_\theta \ln\left[p_{\theta^{(t)}}(y|x, z_{1:K}, e)\right], \\ \text{with } e = \texttt{one\_hot}[\arg\max_k \alpha^{(t+1)}] \;\forall (x, y) \in \mathcal{D} \end{gathered} \tag{8.27}$$

where λ is the learning rate.

Note that the coefficient α is data point dependent, and the derivation of EM algorithms considers a subset of data points $\mathcal{D}$. However, in meta learning scenarios, we handle large-scale datasets, and the above-mentioned EM framework is computationally expensive and impractical. Due to these considerations, *we do not apply EM algorithms to estimate the discrete distribution* and instead variational inference is employed for the assignment latent variable in optimization.

Variational Distributions

For continuous latent variables, diagonal Gaussians are commonly used as variational distributions. With Gaussian variational posteriors $\mathcal{N}(z;\mu,\Sigma)$ and corresponding priors $\mathcal{N}(z;\mu_p,\Sigma_p)$, the Kullback–Leibler Divergence can be analytically computed as follows.

$$D_{KL}[\mathcal{N}(z;\mu,\Sigma) \parallel \mathcal{N}(z;\mu_p,\Sigma_p)] = \frac{1}{2}[\ln\frac{|\Sigma_p|}{|\Sigma|} - d + (\mu-\mu_p)^T\Sigma_p^{-1}(\mu-\mu_p) + \mathtt{Tr}\{\Sigma_p^{-1}\Sigma\}] \tag{8.28}$$

Meanwhile, when it comes to categorical distributions, the corresponding prior distribution is selected as $\mathtt{Cat}(K,\alpha_0)$ with distribution parameters $\alpha_0 = [\alpha_{0,1},\alpha_{0,2},\ldots,\alpha_{0,K}]$. And the Kullback–Leibler Divergence is computed as follows.

$$D_{KL}[\mathtt{Cat}(K,\alpha_*) \parallel \mathtt{Cat}(K,\alpha_0)] = \sum_{k=1}^{K}\alpha_{*,k}\ln\left[\frac{\alpha_{*,k}}{\alpha_{0,k}}\right] \tag{8.29}$$

When $\alpha_0 = [\frac{1}{K},\frac{1}{K},\ldots,\frac{1}{K}]$, the divergence is further simplified as follows.

$$D_{KL}[\mathtt{Cat}(K,\alpha_*) \parallel \mathtt{Cat}(K,\alpha_0)] = \sum_{k=1}^{K}\alpha_{*,k}\ln\left[\frac{\alpha_{*,k}}{1/K}\right] = \sum_{k=1}^{K}\ln\alpha_{*,k} + \ln K \tag{8.30}$$

Lower Bound on the Evidence for Few-Shot Supervised Learning

Since K-expert latent variables are independent in settings, we denote the corresponding variational parameters by $q_{\phi_1} = \{q_{\phi_{1,1}}, q_{\phi_{1,2}}, \ldots, q_{\phi_{1,K}}\}$. $\phi_{1,k}$ denotes parameters of encoders for k-th expert model. Hence, the distribution follows that $q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^C) = \prod_{k=1}^{K} q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^C)$ and $q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T) = \prod_{k=1}^{K} q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^T)$.

Note that $(x,y) \in \mathcal{D}_\tau^T$ and the variational posterior for expert latent variables are $q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)$ in the general NPs. As for the variational posterior for assignment latent variables, we choose $q_{\phi_{2,1}}(e|x,y,z_{1:K})$ as default. We will use these notations to formulate the evidence lower bound (ELBO) as follows.

$$\ln p(y|x,\mathcal{D}_\tau^C) = \ln \int p(y|x,z_{1:K}) p(z_{1:K}|\mathcal{D}_\tau^C) dz_{1:K} \tag{8.31a}$$

$$\geq \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)} \left[\ln p(y|x,z_{1:K})\right] - D_{KL}[q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T) \parallel p(z_{1:K}|\mathcal{D}_\tau^C)] \tag{8.31b}$$

$$= \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)} \left[\ln \sum_{k=1}^{K} p(y, e_k = 1|x,z_{1:K})\right] - D_{KL}[q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T) \parallel p(z_{1:K}|\mathcal{D}_\tau^C)] \tag{8.31c}$$

$$= \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)} \left[\ln \sum_{k=1}^{K} p(y|x,z_k) p(e_k = 1|x,z_{1:K})\right] - D_{KL}[q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T) \parallel p(z_{1:K}|\mathcal{D}_\tau^C)] \tag{8.31d}$$

$$\geq \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)} \left[\mathbb{E}_{q_{\phi_{2,1}}(e|x,y,z_{1:K})} \left[\ln p_\theta(y|x,z_{1:K},e)\right]\right] \tag{8.31e}$$

$$-\mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)} \left[D_{KL}[\underbrace{q_{\phi_{2,1}}(e|x,y,z_{1:K})}_{\text{variational discrete posteriors}} \parallel p(e|x,z_{1:K})]\right] \tag{8.31f}$$

$$-\sum_{k=1}^{K} D_{KL}[\underbrace{q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^T)}_{K \text{ functional experts}} \parallel p(z_k|\mathcal{D}_\tau^C)] \tag{8.31g}$$

$$\approx \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)} \left[\mathbb{E}_{q_{\phi_{2,1}}(e|x,y,z_{1:K})} \left[\ln p_\theta(y|x,z_{1:K},e)\right]\right] \tag{8.31h}$$

$$-\mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)} \left[D_{KL}[\underbrace{q_{\phi_{2,1}}(e|x,y,z_{1:K})}_{\text{variational discrete posteriors}} \parallel \underbrace{p_{\phi_{2,2}}(e|x,z_{1:K})}_{\text{variational discrete priors}}]\right] \tag{8.31i}$$

$$-\sum_{k=1}^{K} D_{KL}[\underbrace{q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^T)}_{K \text{ functional experts}} \parallel q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^C)] = -\mathcal{L}(\theta,\phi_1,\phi_2) \quad \square \tag{8.31j}$$

By introducing the variational distribution q_{ϕ_2} for the discrete assignment latent variable e, Eq.(8.31.d) is further bounded by Eq. (8.31.e-g). Recall that when vanilla NP modules are used here, the approximate posterior in Eq. (8.31) in meta training should be substituted with $q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)$ with the corresponding approximate prior $p(z_k) = q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^C)$. And this matches the general form in the main paper for $-\mathcal{L}(\theta,\phi_1,\phi_2)$ in Eq. (4.7). When Dirac delta distributions are used in MoE-NPs, the divergence term

about the continuous latent variable is removed as default. Denoting the approximate posterior by $q_{\phi_{2,1}}(e|x,y,z_{1:K}) = \texttt{Cat}(e;[\alpha_1(x,y,z_{1:K}),\alpha_2(x,y,z_{1:K}),\ldots,\alpha_K(x,y,z_{1:K})])$, we rewrite the log-likelihood inside the ELBO as Eq. (8.32).

$$\mathbb{E}_{q_{\phi_{2,1}}(e|x,y,z_{1:K})}\left[\ln p(y|x,z_{1:K},e)\right] = \sum_{k=1}^{K} \alpha_k \ln p(y|x,z_k) \tag{8.32}$$

As for the approximate posterior of the assignment latent variable $q_{\phi_{2,1}}(e|x,y,z_{1:K})$, we provide two ways of implementations in our experiments: (i) use the target input y as the additional input to formulate $q_{\phi_{2,1}}(e|x,y,z_{1:K})$ (ii) use the same form as the conditional prior $q_{\phi_{2,1}}(e|x,y,z_{1:K}) = p_{\phi_{2,2}}(e|x,z_{1:K})$.

Selection of Categorical Approximate Posteriors/Priors

As previously observed in Acrobot system identification results, increasing the number of expert latent variables tends to weaken the generalization capability. This also happens in image completion, so we set the number of experts used is 2 in the task. We can attribute this to inference sub-optimality in categorical approximate posteriors/priors.

Remember that in image completion and Acrobot system identification, the used approximate posterior for the categorical latent variable is $q_{\phi_{2,1}}(e|x,y,z_{1:K})$ with the target information y for the input. Since the developed MoE-NP is a VAE-like model (Kingma and Welling, 2013), the number of expert latent variables K decides the dimension of the assignment latent variable e. In auto-encoder models, when the dimension of latent variables in all hidden layers is higher than that of the input, the model tends to copy the input to the output and fails to learn effective representations. This is the direct source of overfitting and applies to conditional VAE methods (Sohn et al., 2015). For example, the output dimension in Acrobot is 6, which implies the bottleneck constraint is weaker when the number of experts exceeds 6.

It is reasonable to alleviate such sub-optimality by directly using the conditional prior as the approximate posterior $q_{\phi_{2,1}}(e|x,y,z_{1:K}) = p_{\phi_{2,2}}(e|x,z_{1:K})$. You can find more clues from the following stochastic gradient estimates for the assignment latent variables in Eq. (8.36). Meanwhile, we report empirical results in image completion when $q_{\phi_{2,1}}(e|x,y,z_{1:K}) = p_{\phi_{2,2}}(e|x,z_{1:K})$ in Sec. (B.8.1). Moreover, you can see that inference in this way does not suffer the overfitting issue caused by more experts.

Stochastic Gradient Estimates

Here the stochastic gradient estimates with respect to parameters in the negative ELBO $\mathcal{L}(\theta,\phi_1,\phi_2)$ in Eq. (4.7) are provided as follows.

$$\frac{\partial}{\partial\theta}\mathcal{L}(y;x,\theta,\phi_1,\phi_2) = \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)} \sum_{k=1}^{K} q_{\phi_{2,1}}(e_k=1|x,y,z_{1:K}) \frac{\partial}{\partial\theta} \ln p_\theta(y|x,z_k) \tag{8.33}$$

$$\begin{aligned}\frac{\partial}{\partial\phi_{1,k}}\mathcal{L}(y;x,\theta,\phi_1,\phi_2) = &\int\left[\frac{\partial}{\partial\phi_{1,k}}q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^T)\right]\ln p_\theta(y|x,z_k)dz_k \\ &-\frac{\partial}{\partial\phi_{1,k}}D_{KL}[q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^T)\parallel q_{\phi_{1,k}}(z_k|\mathcal{D}_\tau^C)]\end{aligned} \tag{8.34}$$

$$\begin{aligned}\frac{\partial}{\partial\phi_2}\mathcal{L}(y;x,\theta,\phi_1,\phi_2) = &\mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)}\left[\sum_{k=1}^{K}\left[\frac{\partial}{\partial\phi_2}q_{\phi_{2,1}}(e_k=1|x,y,z_{1:K})\right]\ln p_\theta(y|x,z_k)\right] \\ &-\mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)}\left[\frac{\partial}{\partial\phi_2}D_{KL}[q_{\phi_{2,1}}(e|x,y,z_{1:K})\parallel p_{\phi_{2,2}}(e|x,z_{1:K})]\right]\end{aligned} \tag{8.35}$$

The reparameterization trick (Kingma and Welling, 2013) is used to sample values from variational distributions of expert latent variables throughout the inference process and stochastic gradient estimates in Eq. (8.33)/(8.34)/(8.35). In prediction processes, the way to get values of assignment latent variables follows that in (Gumbel, 1954).

Besides, we provide the stochastic gradient estimate for another case when the variational posterior for the assignment latent variable is selected as the variational prior, which means $q_{\phi_{2,1}}(e|x,y,z_{1:K}) = p_{\phi_{2,2}}(e|x,z_{1:K})$. This case can drop off the divergence term for the discrete variable. Let the conditional prior for the discrete variable be $p_{\phi_{2,2}}(e|x,z_{1:K}) = \texttt{Cat}(e;[\alpha_1(x,z_{1:K}),\alpha_2(x,z_{1:K}),\ldots,\alpha_K(x,z_{1:K})])$, we apply the log-derivative trick in a REINFORCE estimator (Williams, 1992) to Eq. (8.35) and can obtain the following equation as the gradient estimator[5].

$$\begin{aligned}\frac{\partial}{\partial\phi_{2,2}}\mathcal{L}(y;x,\theta,\phi_1,\phi_2) &= \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)}\left[\sum_{k=1}^{K}\left[\frac{\partial}{\partial\phi_{2,2}}p_{\phi_{2,2}}(e_k=1|x,z_{1:K})\right]\ln p_\theta(y|x,z_k)\right] \\ &= \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)}\left[\sum_{k=1}^{K}p_{\phi_{2,2}}(e_k=1|x,z_{1:K})\left[\underbrace{\frac{\partial}{\partial\phi_{2,2}}\ln p_{\phi_{2,2}}(e_k=1|x,z_{1:K})\ln p_\theta(y|x,z_k)}_{\text{Score Function}}\right]\right] \\ &= \mathbb{E}_{q_{\phi_1}(z_{1:K}|\mathcal{D}_\tau^T)}\left[\sum_{k=1}^{K}\alpha_k\frac{\partial}{\partial\phi_{2,2}}\ln p_{\phi_{2,2}}(e_k=1|x,z_{1:K})\ln p_\theta(y|x,z_k)\right]\end{aligned} \tag{8.36}$$

As can be seen from Eq. (8.36), the posterior update implicitly exploits supervision information.

5 For discrete latent variables, we can obtain the analytical form of the stochastic gradient.

Estimates of Statistics

Momentum. Given the pre-trained MoE-NPs, we can formulate the statistical momentum in predictive distributions. For the first order momentum, equivalently mean of the predictive distribution, we need to compute the conditional version $\mathbb{E}[Y|X=x,\mathcal{D}_\tau^C]$ in meta learning scenarios. Here the predictive distribution of one expert is parameterized in the form $p(y|x,z_k)=\mathcal{N}(y;m(x,z_k),\Sigma_k)$, where m is the learned mean function using a neural network and σ^2 is a variance parameter. Using a single stochastic forward pass in the expert latent variable $z_{1:K}$, we can derive the estimate of the predictive mean $\hat{m}=\mathbb{E}[Y|X=x,\mathcal{D}_\tau^C]$.

$$\hat{m}=\sum_{k=1}^{K}\alpha_k\cdot m(x,z_k),\ \alpha_k=p_{\phi_{2,2}}(e_k=1|z_{1:K},x) \tag{8.37}$$

The second order moment can be estimated accordingly. Here we consider the case when the output is one dimensional and $\Sigma_k=\sigma_k^2$.

$$\mathbb{V}[Y|X=x,\mathcal{D}_\tau^C]=\mathbb{E}[Y^2]-\mathbb{E}[Y]^2=\sum_{k=1}^{K}\alpha_k(\sigma_k^2+m(x,z_k)^2)-\hat{m}^2 \tag{8.38}$$

Entropy. Note that our developed MoE-NPs can also be applied to out of detection (O.O.D) tasks. In this case, the entropy of predictive distribution plays a crucial role. Though the exact estimate of the predictive entropy for MoE-NPs is intractable due to the complexity inside the mixture components, we can measure the expected result of the entropy $\mathbb{E}[\mathbb{H}(Y)|X=x,\mathcal{D}_\tau^C]$ in prediction. We still use a single stochastic forward pass in the expert latent variable $z_{1:K}$ in estimation. If $\mathbb{H}(Y_k|X=x,z_k)=-\int p(Y=y|X=x,z_k)\ln p(Y=y|X=x,z_k)dY$ is bounded $\forall k\in\{1,\ldots,K\}$, the estimate of entropy term is as follows.

$$\begin{aligned}\hat{\mathbb{E}}[\mathbb{H}(Y)|X=x,\mathcal{D}_\tau^C]&=\sum_{k=1}^{K}\alpha_k\int p(Y=y|X=x,z_k)\mathbb{H}(Y=y)dy\\&=\sum_{k=1}^{K}\alpha_k\mathbb{E}[\mathbb{H}(Y_k|X=x,z_k)]\end{aligned} \tag{8.39}$$

B.7 *Experimental Settings and Neural Architectures*

In this section, we provide with more experimental details. Importantly, neural modules of MoE-NPs in the PyTorch version are listed. For the few-shot regression, we provide an example of our implementation of MoE-NPs from the anonymous Github link `https://github.com/codeanonymous233/ICMoENP`. For the context-based meta RL algorithms, you can find the implementation of MoE-NPs from the anonymous Github link `https://github.com/codeanonymous233/MoENP`.

Dataset & Environments

System Identification. Note that in the used Acrobot simulator [6], the observation is the preprocessed state as a 6 dimensional vector $[\sin(\theta_1), \cos(\theta_1), \sin(\theta_2), \cos(\theta_2), \theta'_1, \theta'_2]$. The input of the Acrobot system is the concatenation of the observation and the executed action $[\sin(\theta_1), \cos(\theta_1), \sin(\theta_2), \cos(\theta_2), \theta'_1, \theta'_2, a]$. The output of Acrobot system is the predicted transited state. We generate 16 meta training tasks by varying the masses of two pendulums m_1 and m_2, which means the hyper-parameters of the system come from the Cartesian combination of the set $m_1 \in \{0.75, 0.85, 0.95, 1.15\}$ and $m_2 \in \{0.75, 0.85, 0.95, 1.15\}$. In meta training processes, a complete random policy interacts with a batch of sampled MDPs to formulate a transition dataset. As for meta testing tasks, we follow the same way to generate tasks by setting $m_1 \in \{0.85, 1.05, 1.25\}$ and $m_2 \in \{0.85, 1.05, 1.25\}$.

Image Completion. CIFAR10 dataset consists of 60000 32x32 color images in 10 categories. Among these images, 50000 are for meta training, with the rest for meta testing as the default in image completion tasks. CIFAR10 images are processed via torchvision modules to normalize the pixel values between $[0, 1]$.

Environments in Meta Reinforcement Learning. Note that 2-D point robot navigation tasks, the distribution for meta training is a mixture of uniform distributions $[0, 2\pi/12] \cup [5\pi/12, 7\pi/12] \cup [10\pi/12, \pi]$. The rest of the regions along the arc is for out of distribution tasks. The tasks in Mujoco (Todorov et al., 2012) follow adaptations from (Rakelly et al., 2019; Ren et al., 2019), where goals/velocities or multiple hyper-parameters of simulation systems are sampled from mixture distributions. The horizon of an episode for a mixture of point robots is 20, while that for Mujoco environments is 500. The required number of meta training processes environment steps are $2.5 * 1e6$ for point robots, $7.5 * 1e6$ for Half-Cheetah-CD, and $6.5 * 1e6$ for Slim-Humanoid-CG. We leave more details and settings of each environment in the above github code link. 2-D point robots attempt to reach goals located in specified arc regions. 2-D Cheetah robots aim at running in given directions. The task includes multiple target directions, and these change with split steps of episodes. 3-D Humanoid robots aim at running towards goals. The task includes multiple goals, and these change with split steps of episodes.

Implementations in Meta Learning Tasks

Toy Experiments. The input of functions is in a range $[-\pi, \pi] \cup [\pi, 2\pi]$. The general implementations of MoE-NPs are as follows. The x-domain for the function $f_1(x) = \sin(x) + \epsilon_1$ with $\epsilon_1 \sim \mathcal{N}(0, 0.03^2)$ is $[-\pi, \pi]$, while that for $f_2(x) = \cos(x) + \epsilon_2$ with $\epsilon_2 \sim \mathcal{N}(0, 0.01^2)$ is $[\pi, 2\pi]$. Sampling from these two components leads to a mixture dataset. The `Encoder` for all continuous latent variables is with two hidden layers (32 neuron units each). The Gaussian distribution is used for continuous latent variables in MoE-NPs. The `Encoder` for the discrete assignment latent variable in MoE-NPs

6 https://github.com/openai/gym/blob/master/gym/envs/classic_control/acrobot.py

corresponds to a softmax-output neural network with two hidden layers (16 neuron units each). The `Decoder` also has three hidden layers (128 neuron units each). The number of data points in each sampled task is 100. For gradient based methods, we use implementations of MAML[7] and CAVIA[8].

System Identification. The general implementations are as follows. For all NP related models, the dimension of a latent variable is set to be 16. The `Encoder` for all continuous latent variables is with two hidden layers (32 neuron units each). Since this case works best in few-shot supervised learning, Dirac delta distributions are used for continuous latent variables in MoE-NPs. For MoE-NPs, we use three expert latent variables as the default, and the `Encoder` for the discrete assignment latent variable in MoE-NPs corresponds to a softmax-output neural network with two hidden layers (32 neuron units each). The `Decoder` also has four hidden layers (200 neuron units each). The number of tasks in batch training is 4, batch size in training is 200 transition steps (for each task). The horizon for each episode of transitions is 200 time steps. In each iteration of meta training, 4 different environments are randomly selected, and the uniform random controller is used to interact for the collection of 5 episodes (for each task). In training dynamics systems, the training batch size in the transition buffer is 200, the training epoch is 5, and the whole process iterates until convergence (the iteration number is 25). The learning rate for Adam optimizer is $1e-3$ as the default.

Image Completion. The general implementations follow that in (Zintgraf et al., 2019; Garnelo et al., 2018a) and are applied to all baselines and MoE-NPs. The dimension of a latent variable is set to 128. The `Encoder` for all NP variants is with three hidden layers (128 neuron units each). Since this case works best in few-shot supervised learning, Dirac delta distributions are used for continuous latent variables in MoE-NPs. The `Decoder` also has five hidden layers (128 neuron units each). For MoE-NPs, we use three expert latent variables as the default, and the `Encoder` for the discrete assignment latent variable corresponds to a softmax-output neural network with two hidden layers (32 neuron units each). Adam (Kingma and Welling, 2013) is used as the optimizer, where the learning rate is set to be $5e-4$. The batch size in training is 8 images, and we meta train the model until convergence (the maximum epoch number is 50). Also, note that the number of context pixels in CAVIA is 10 for fast adaptation in default implementations, which leads to the best testing result of CAVIA in 10 pixel cases in Fig. (10). Note that, to train NP models, including CNP/NP/FCRL/MoE-NP, we set the form of negative log-likelihood objective consistent based on that in (Le et al., 2018). But in evaluation, to keep results of all methods consistent, we follow that in (Garnelo et al., 2018a; Zintgraf et al., 2019; Gondal et al., 2021) and report the MSEs in Fig. (10) in the testing phase.

7 `https://github.com/cbfinn/maml_rl`
8 `https://github.com/lmzintgraf/cavia`

Meta Reinforcement Learning. In terms of implementations of baselines, we directly use the following open sourced code: PEARL[9], MAML[10] and CAVIA[11]. Note that the TRPO algorithm (Schulman et al., 2015) is used for MAML/CAVIA as the default. We do not change too much except the replacement of our environments when running experiments.

Further, we provide more details on how to modify MoE-NPs in meta RL domains. Notice that MoE-NP can also be seen as a latent variable model, and there exists a close relationship with PEARL algorithms (Rakelly et al., 2019) when it comes to meta RL. Implementations of MoE-NPs are the same as in PEARL (Rakelly et al., 2019) except for latent variable distributions and the inference way. Note that Soft Actor Critic (SAC) algorithm (Haarnoja et al., 2018) is used in policy optimization, which requires parameterization of both actor and critic functions. As for the number of experts in MoE-NPs, we use 3 for all environments as default. You can find more details about neural architectures/optimizers for each module from the link mentioned above.

As mentioned before, we use $p(z_k) = \mathcal{N}(0, I)$ as the prior distribution for expert latent variables. The approximate posterior is parameterized with a diagonal Gaussian distribution. The coefficient for KL divergence terms in Eq. (4.14) are $\beta_0 = 1.0, \beta_1 = 1.0$. The main paper finds the meta-training processes for reinforcement learning in Algorithm (4). In terms of meta-testing processes, we report the pseudo code in Algorithm (5).

B.8 *Additional Experimental Results*

Additional Analysis of Learned Latent Variables

Here we give more detailed analysis *w.r.t.* learned latent variables in MoE-NPs.

Number of Expert Latent Variables. By setting the approximate posterior of assignment latent variables as $q_{\phi_{2,1}}(e|x, y, z_{1:K}) = p_{\phi_{2,2}}(e|x, z_{1:K})$, we further investigate the scalability issue of MoE-NPs in CIFAR10 image completion. As reported in Table (18), we can find with more experts (>= 3), the performance can be further improved, and no overfitting issue occurs. It can also be inferred that when the number of experts reaches a certain level, the improvement from the increase of expert numbers is quite limited. So, in general, when the output dimension is lower, the best choice of the approximate posterior for assignment latent variables is in a form without y as the input.

Additional Results of NP Variants in Toy Regression

Note that the variation of tasks in the previous toy regression is quite limited, and its goal is to show the role of latent variables. To further examine the performance, we construct the mixture of sinusoidal functions by varying the amplitude and phase.

The learning data points are randomly sampled in x-domain and merged from a mixture of randomized functions $f_1(x) = A\sin(x - B) + \epsilon$ in x-domain $[-\pi, \pi]$ and $f_2(x) =$

9 https://github.com/katerakelly/oyster
10 https://github.com/cbfinn/maml_rl
11 https://github.com/lmzintgraf/cavia

Table 18: Pixel-wise mean squared errors (MSEs) with varying number of experts in CIFAR10 image completion. The number of random context points is varied in a range $(10, 200, 500, 800, 1000)$ to test performance at different levels.

#	10	200	500	800	1000
MoE-NPs (3 experts)	0.0482	0.0183	0.0170	0.0166	0.0165
MoE-NPs (5 experts)	0.0362	0.0103	0.0071	0.0061	0.0057
MoE-NPs (7 experts)	0.0359	0.0095	0.0062	0.0052	0.0048

$A\cos(x-B)+\epsilon$ in x-domain $[\pi, 3\pi]$ with equal probability, where $\epsilon \sim \mathcal{N}(0, 0.03^2)$. The range of the amplitude is $A \in [0.1, 5.0]$ while that of the phase is $B \in [0, \pi]$.

We sample a batch of data points in each training iteration and randomly partition context points and target points for learning. Each task consists of 100 randomly sampled data points from the mixture of sinusoidal functions with the random number of context points between $[5, 50]$. The default number of tasks in a meta batch is 25, and we set the number of iteration steps at most 50000. As for neural architectures of all baselines, we retain that in the previous toy regression in Sec. (B.7.1). Still, we use two experts for MoE-NPs as default to fit mixture sinusoidal functions.

Table 19: Test Performance in Mixture Sinusoidal Functions. Shown are mean square errors and standard deviations in fitting 500 sampled tasks. The best results in 5 runs are in bold with standard deviations in bracket.

CNP	NP	FCRL	AttnNP	MoE-NP
0.053±1E-4	0.070±3E-4	0.040±0.0	**0.027±2E-4**	0.032±1E-4

In the meta testing phase, we draw up 500 tasks with 15 random data points selected as the context. These testing tasks are generated in the way: the couple of the amplitude and the phase $[A, B]$ are orderly set from the amplitude list $\texttt{numpy.linspace}(0.1, 5.0, \texttt{num} = 500)$ and phase list $\texttt{numpy.linspace}(0.0, \pi, \texttt{num} = 500)$. The sampled x-values for these tasks are a list $\texttt{torch.linspace}(-\pi, 3\pi, \texttt{steps} = 500)$. As shown in Table (19), AttnNP achieves the best performance in meta testing, followed by MoE-NP.

Additional Results of NP Variants in System Identification

To understand how performance evolves with more context transitions in Acrobot system, we extend the result of 50 context points in the main paper to Fig. (27). As can be seen, MoE-NP still outperforms all NP baselines in all cases. Moreover, with the increase in context transitions, we can find that predictive MSEs degrade accordingly.

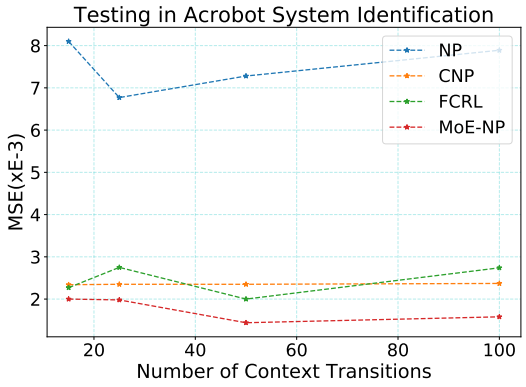


Figure 27: Asymptotic Performance in System Identification of NPs Family. The numbers of random transitions as the context are respectively $\{15, 25, 50, 100\}$.

Comparison with Attentive Neural Processes

Since neural architectures for attentive neural processes (AttnNPs) (Kim et al., 2019a) are a bit different from used baselines and cannot be trivially modified to meta RL cases, we report additional results in this subsection.

We implement AttnNPs with dot attention networks (since AttnNPs have more model complexity and can easily lead to cuda out of memory in practice, we choose the basic version of AttnNPs), the input embedding dimension of to compute attention weights is 32 and 4 layers are used to transform the deterministic embedding z_{attn}. The local embedding is concatenated with the input x and the global latent variable z for the decoder. We use three heads for system identification tasks and one head for image completion tasks. The related results are reported as follows. It can be seen in Table (20), MoE-NPs still outperform AttnNPs, while AttnNPs beat NPs a lot in predicting Acrobot system dynamics. As for CIFAR10 image completion, we can conclude in Table (21) that Mixture Expert inductive biases are more effective than local latent variables embedded in attention modules.

Table 20: System identification in Acrobot. Meta testing results are reported. We use the number of random transitions as the context and test performance for AttnNPs to compare. Figures in the Table are scaled by multiplying `E-3` for means and standard deviations.

#	15	25	50	100
AttnNP	3.0±0.36	2.8±0.41	2.5±0.14	2.8±0.19
MoE-NP	**2.0±0.45**	1.9±**0.28**	**1.4±0.06**	**1.5±0.06**

Augmenting MoE-NPs with Convolutional Modules

In this section, we examine the chance of Since neural architectures of encoders are quite different between ConvCNPs (Gordon et al., 2020a) and previously mentioned NP baselines, we only report the results of NP related models with the same functional

Table 21: Pixel-wise mean squared errors (MSEs) in the image completion tasks on the CIFAR10 dataset. The number of random context points is varied in a range $(10, 200, 500, 800, 1000)$ to test performance at different levels.

#	10	200	500	800	1000
AttnNP	0.0377	0.0223	0.0217	0.0215	0.0215
MoE-NP	**0.0377**	**0.0142**	**0.0117**	**0.0110**	**0.0107**

encoder structures in the main paper. Note that the translation equivariance is injected in ConvCNPs, which is a strong inductive bias for the image dataset. Naturally, we also develop the convolutional version of MoE-NPs. Moreover, we report the different results in image completion here. In our settings, MoE-ConvCNPs use 3 experts in convolutional modules. The implementation of ConvCNPs can be found in (Gordon et al., 2020a). It can be seen that in Table. (22), in comparison to ConvCNP, the image completion performance is further improved with the help of multiple experts.

Table 22: Pixel-wise mean squared errors (MSEs) in the image completion tasks on the CIFAR10 dataset. The number of random context points is varied in a range $(10, 200, 500, 800, 1000)$ to test performance at different levels.

#	10	200	500	800	1000
ConvCNPs	0.036	0.0062	0.002	0.0011	0.0019
MoE-ConvCNPs	**0.035**	**0.0057**	**0.0017**	**0.0007**	**0.0009**

More Visualization Results

Here we include more visualized CelebA images by varying the number of observed pixels. These are produced using CNN Augmented MoE-NPs (MoE-ConvCNPs). Fig.s (28)/(29)/(30)/(31) are image completion results given the fixed number of random context pixels. Fig.s (32)/(33)/(34)/(35) are image completion results given the fixed number of ordered context pixels.

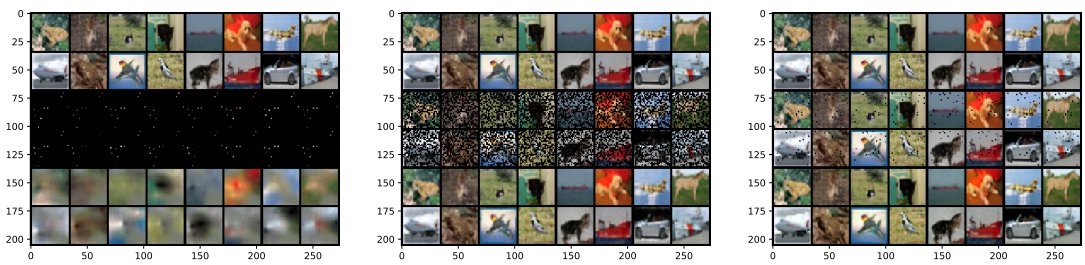


Figure 28: Image Completion Results using CNN Augmented MoE-NPs.

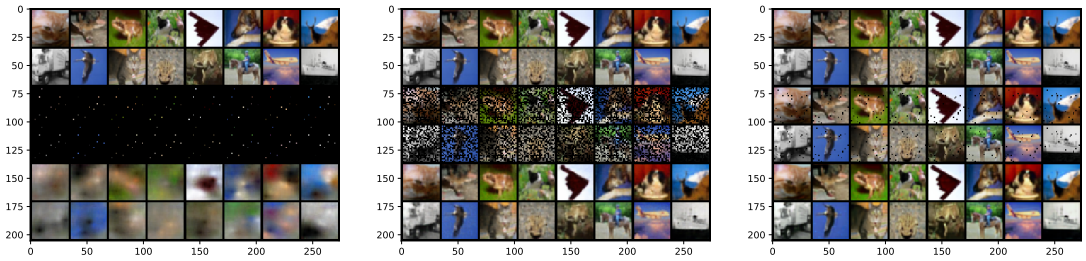


Figure 29: Image Completion Results using CNN Augmented MoE-NPs.

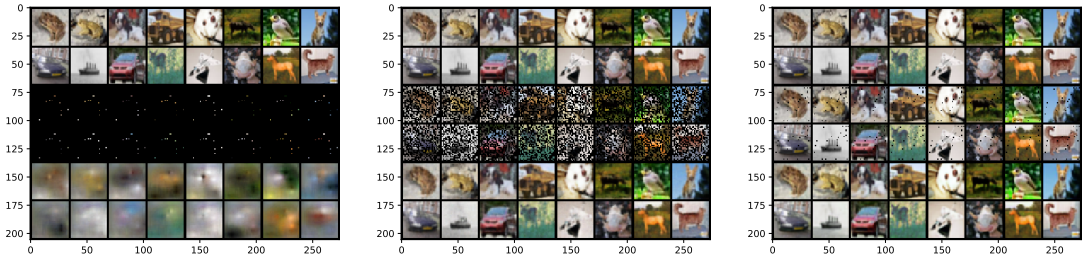


Figure 30: Image Completion Results using CNN Augmented MoE-NPs.

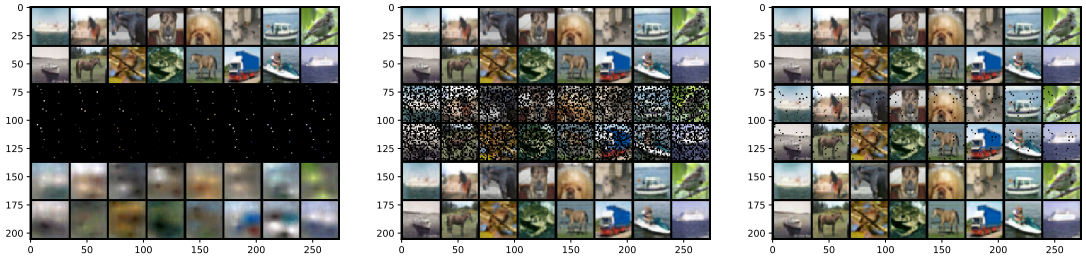


Figure 31: Image Completion Results using CNN Augmented MoE-NPs.

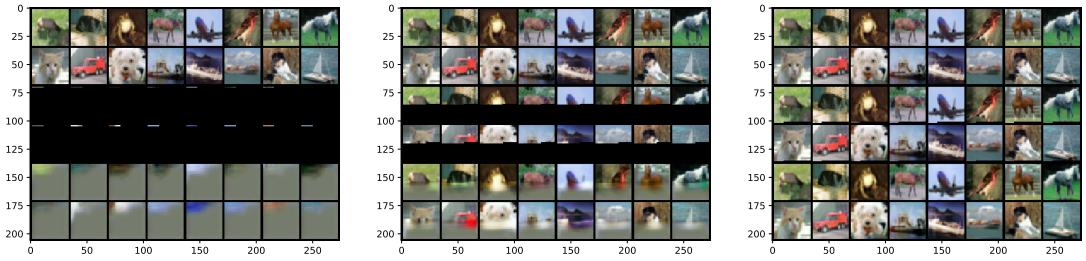


Figure 32: Image Completion Results using CNN Augmented MoE-NPs.

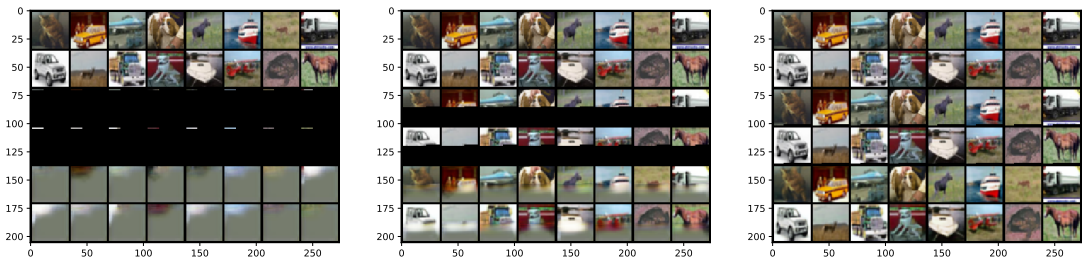


Figure 33: Image Completion Results using CNN Augmented MoE-NPs.

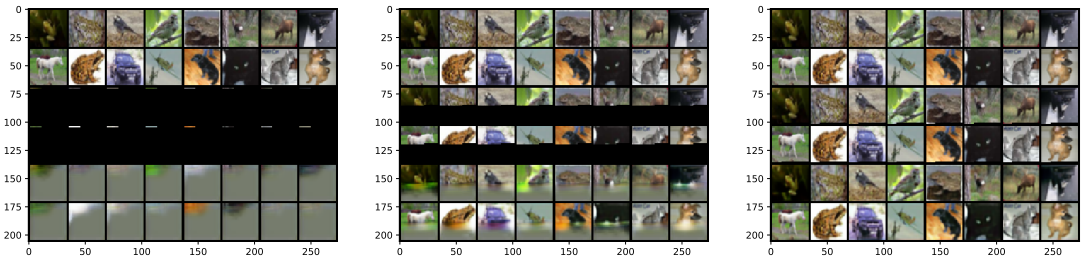


Figure 34: Image Completion Results using CNN Augmented MoE-NPs.

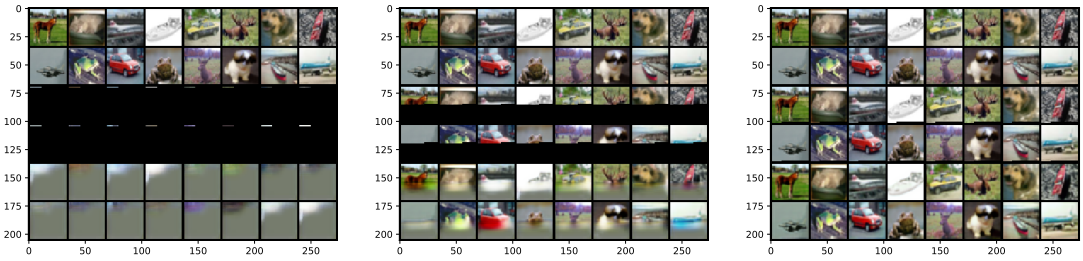


Figure 35: Image Completion Results using CNN Augmented MoE-NPs.

Table 23: Summary of Typical MBMRL Models. Encoders are for dynamics models. Policy search strategies include model predictive control (MPC), policy gradient (PG) methods and amortized policy search (APS). As for fast adaptation, we consider whether this step is directly included in learning dynamics models or policy search.

MBMRL	Encoders	Policy Search	Dynamics Model	Fast adaptation
L2A	NULL	MPC	MAML	DM
MLSM-v0	$q_\phi(z\|\texttt{MeanPool}([x_c, y_c]))$	PG	LVM	DM
MLSM-v1	$q_{\phi_1}(z\|\texttt{MeanPool}([x_c, y_c]))$ $f_{\phi_2}(z_t\|\texttt{Attn}([x_c, y_c], x_t))$	PG	LVM	DM
GSSM	$q_{\phi_2}(z_c\|\texttt{GNN}([x_c, y_c]))$ $q_{\phi_1}(z_t\|\texttt{GNN}([x_c, y_c], x_t))$	PG	LVM	DM
GSSM+APS	$q_{\phi_2}(z_c\|\texttt{GNN}([x_c, y_c]))$ $q_{\phi_1}(z_t\|\texttt{GNN}([x_c, y_c], x_t))$	APS	LVM	DM/PS

Table 24: Parameter Scales of Context-based Dynamics Models in different tasks. It can be found GSSM and MLSM-v0 have the same model complexity while MLSM-v1 has more parameters in meta dynamics models.

	MLSM-v0	MLSM-v1	GSSM
Cart-Pole	8.9*1E4	9.3*1E4	8.9*1E4
Acrobot	8.1*1E5	8.4*1E5	8.1*1E5
H-Cheetah	2.1*1E5	2.3*1E5	2.1*1E5
S-Humanoid	2.3*1E5	2.6*1E5	2.3*1E5

C SUPPLEMENTARY MATERIALS IN CHAPTER 5

C.1 *Frequently Asked Questions*

According to feedback from other reviewers, we summarize frequently asked questions and add explanations in this part. This is to make our work clearer to potential readers.

Possibility to combine Graph Neural Net (GNN) modules with meta model-free RL methods. A combination of GNN latent variable and meta model-free methods in our work is not applicable for an ablation experiment. That is because (1) the input and output for GNN modules cannot be accordingly specified (in model-based settings, the input and output are respectively $[s, a]$ and Δs; in model-free settings, the input is s for policies). (2) the optimization objectives for GNN modules differ a lot in model-based (Maximize the likelihood of dynamics prediction in GNN related modules) and model-free cases (Maximize the expected rewards in GNN related modules). (3) we do

Table 25: Mean Square Errors (MSEs) in Meta-testing Tasks using Learned Dynamics Models. (For each testing task, 50 episodes are sampled to average. Figures in brackets are standard deviations across testing tasks, with bold ones the best.) Here the results for Acrobot are different from that in MoE-NPs since we use different neural architectures in separate research periods and MoE-NPs come out later in publications.

Env	GSSM	M-DPILCO	MLSM-v0	MLSM-v1	L2A
Cart-Pole	**0.0296±0.042**	0.0475±0.051	0.0626±0.081	0.0310±0.036	0.0397±0.04
Acrobot	**0.0024±0.0019**	0.0030±0.0029	0.004±0.0042	**0.0024±0.0021**	0.0039±0.0017
H-Cheetah	**0.530±0.22**	0.678±0.14	0.533±0.14	0.636±0.14	0.785±0.084
S-Humanoid	1.78±0.13	1.9±0.15	2.0±0.16	**1.75±0.15**	2.364±0.078

not find an appropriate GNN related meta model-free RL baseline, and the extension is non-trivial to design.

Technical summary of context-based meta model-based RL methods. Comparisons between our developed graph structured surrogate model and other dynamics models are summarized in Table (23). We mainly focus on methods to enable fast adaptation in both dynamics models and policy networks.

Performance comparison to other meta model-based RL algorithms with non-GNN encoders. In Chapter (5) Section (5.5), non-GNN encoders correspond to NPs in MLSM, where a mean reduction is already used to obtain the encoded latent variable. Recurrent encoders are improper in our settings since the randomly sampled transitions from the memory buffer to identify the dynamical system are not entirely sequential in the dataset. However, the use of Recurrent encoders is more effective when the transitions in the dynamics buffer are collected and stored in an ordered way. The model benefits from the sequential information. In this case, CaDM (Lee et al., 2020b) can achieve SOTA performance in the domain.

Performance comparison to other existing meta model-free RL methods, *e.g.* RL2 (Duan et al., 2016). See Chapter (5) Section (5.5), PE-PPO follows the same implementation in PEARL (Rakelly et al., 2019) except that PPO is used in policy optimization. This is to ensure all policy optimization methods are consistent in experimental analysis. We also tried PEARL in model-free experiments with 1x volume of samples, but the results were poorer than used baselines. Meanwhile, please refer to learning curves of other model-free meta RL papers with 1x volume of samples, *e.g.*, FOCAL (Li et al., 2020) and MBML (Li et al., 2019), conclusions are: with limited episodes (1x samples), model-free ones, including RL2 or Learn2Learn, work far worse than model-based baselines illustrated in our papers. This means the selection of model-free meta RL baselines does not influence the comparison results, and this is due to the performance bottleneck of model-free ones with limited training episodes.

C.2 *Proof of Lemma 1*

This proof is based on partial results in (Rajeswaran et al., 2020), and an extension is performed here. The context is set in distribution over MDPs $M \sim p(M)$, and a sampled real environment and the corresponding learned dynamics model are respectively denoted as M and $\hat{M}$.

Corollary 1 *Assuming a single step reward in a Markov Decision Process has a supremum value $\mathcal{R}_{\max}$ and the discounted factor for rewards $\gamma < 1$, then the state value function $V^{\pi}(s)$ under a policy π can be bounded with the following inequality.*

$$\max_{s\in\mathcal{S}} V^{\pi}(s) \leq \frac{\mathcal{R}_{\max}}{1-\gamma}, \quad \forall\pi \tag{8.40}$$

Proof 1 *The state value $V^{\pi}(s)$ can be computed in the form of $\mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}_t|S_0 = s\Big] = \int p(\tau)\mathcal{R}(\tau)d\tau$, where the cumulative reward for trajectory τ is $\mathcal{R}(\tau) = \sum_{t=0}^{\infty}\gamma^t\mathcal{R}_t$ with an initial state $S_0 = s$. Also note that $\sup\{\mathcal{R}_t\} \leq \mathcal{R}_{\max}$, it is trivial to verify the equation.*

$$\begin{aligned} \mathcal{R}(\tau) &\leq (\sum_{t=0}^{\infty}\gamma^t)\mathcal{R}_{\max} = \frac{\mathcal{R}_{\max}}{1-\gamma} \\ V^{\pi}(s) &= \int p(\tau)\mathcal{R}(\tau)d\tau \leq \frac{\mathcal{R}_{\max}}{1-\gamma} \end{aligned} \tag{8.41}$$

Note the Bellman equation in terms of any state value $V_{\pi}^{M}(s)$ under a policy π in a dynamics model M,

$$\begin{aligned} V_{\pi}^{M}(s) &= \int \Big(r(s,a,s') + \gamma V_{\pi}^{M}(s')\Big)\pi(a|s)p(s'|s,a)dads' \\ &= \mathcal{R}_{\pi}^{M}(s) + \gamma\mathbb{E}_{s'\sim p_{\pi}^{M}(\cdot|s)}\Big[V_{\pi}^{M}(s')\Big] \end{aligned} \tag{8.42}$$

where $\mathcal{R}_{\pi}^{M}(s)$ is the expected one step rewards and $p_{\pi}^{M}(\cdot|s)$ is the state transition distribution and both depend on the environment and the policy.

Hence, we can naturally estimate the difference between state values in the two mentioned dynamics models with the help of **Corollary** (1) as follows.

$$\begin{aligned} |V_{\pi}^{M}(s) - V_{\pi}^{\hat{M}}(s)| &\leq |\mathcal{R}_{\pi}^{M}(s) - \mathcal{R}_{\pi}^{\hat{M}}(s)| + \gamma|\mathbb{E}_{s'\sim p_{\pi}^{M}(\cdot|s)}\Big[V_{\pi}^{M}(s')\Big] - \mathbb{E}_{s'\sim p_{\pi}^{\hat{M}}(\cdot|s)}\Big[V_{\pi}^{\hat{M}}(s')\Big]| \\ &\leq 2\mathcal{R}_{\max}D_{\mathrm{TV}}[p_{\pi}^{\hat{M}}(\cdot|s), p_{\pi}^{M}(\cdot|s)] + \gamma|\mathbb{E}_{s'\sim p_{\pi}^{M}(\cdot|s)}\Big[V_{\pi}^{M}(s')\Big] - \mathbb{E}_{s'\sim p_{\pi}^{\hat{M}}(\cdot|s)}\Big[V_{\pi}^{M}(s')\Big]| \\ &\quad + \gamma|\mathbb{E}_{s'\sim p_{\pi}^{\hat{M}}(\cdot|s)}\Big[V_{\pi}^{M}(s')\Big] - \mathbb{E}_{s'\sim p_{\pi}^{\hat{M}}(\cdot|s)}\Big[V_{\pi}^{\hat{M}}(s')\Big]| \\ &\leq 2\mathcal{R}_{\max}D_{\mathrm{TV}}[p_{\pi}^{\hat{M}}(\cdot|s), p_{\pi}^{M}(\cdot|s)] + 2\gamma\Big(\max_{s'} V_{\pi}^{M}(s')\Big)D_{\mathrm{TV}}[p_{\pi}^{\hat{M}}(\cdot|s), p_{\pi}^{M}(\cdot|s)] \\ &\quad + \gamma\max_{s'}|V_{\pi}^{M}(s') - V_{\pi}^{\hat{M}}(s')|, \quad \forall s\in\mathcal{S} \end{aligned} \tag{8.43}$$

Since the left side term is satisfied for all states, we can naturally have the following equation.

$$(1-\gamma)\max_{s'}|V_\pi^M(s')-V_\pi^{\hat{M}}(s')| \le 2\Big(\mathcal{R}_{\max}+\gamma\Big(\max_{s'}V_\pi^M(s')\Big)\Big)D_{\mathrm{TV}}[p_\pi^{\hat{M}}(\cdot|s),p_\pi^M(\cdot|s)] \tag{8.44}$$

Then by imposing $\mathbb{E}_{M\sim p(M)}$ over both sides in Eq. (8.44) and with the meta dynamics model approximated error $\mathbb{E}_{\substack{M\sim p(M)\\(s,a)\sim\nu(s,a)}}\Big[D_{\mathrm{TV}}[\hat{P}_M(\cdot|s,a),P_M(\cdot|s,a)]\Big]\le\epsilon$, we can give the regret bound as follows.

$$\mathbb{E}_{M\sim p(M)}\Big[\max_{s'}|V_\pi^M(s')-V_\pi^{\hat{M}}(s')|\Big]\le\frac{2\epsilon\mathcal{R}_{\max}}{(1-\gamma)^2},\quad\forall\pi \tag{8.45}$$

Finally, the performance gap can be measured with Eq. (8.46), and **Lemma** (1) is proved.

$$\mathbb{E}_{M\sim p(M)}\Big[|\mathcal{J}_{\hat{M}}(\pi)-\mathcal{J}_M(\pi)|\Big]\le\frac{2\epsilon\mathcal{R}_{\max}}{(1-\gamma)^2},\quad\forall\pi \tag{8.46}$$

C.3 *Proof of Model Discrepancy*

Here let us refer to optimal policies in an arbitrary MDP $\mathcal{M}$ and its approximation $\hat{\mathcal{M}}$ as π_M and $\pi_{\hat{M}}$ respectively. With the induction in **Lemma** (1), we reuse Eq. (8.44) and it is trivial to verify the following equations.

$$|\mathcal{J}_{\hat{M}}(\pi)-\mathcal{J}_M(\pi)|\le\frac{2\mathcal{R}_{\max}D_{\mathrm{TV}}[p_\pi^{\hat{M}}(\cdot|s),p_\pi^M(\cdot|s)]}{(1-\gamma)^2},\quad\forall\pi \tag{8.47}$$

Hence, we can have the inequality based on the truth that $\pi_{\hat{M}}$ is optimal in $\hat{\mathcal{M}}$ and reuse Eq. (8.44).

$$\begin{aligned}\mathcal{J}_M(\pi_M)&\le\mathcal{J}_{\hat{M}}(\pi_M)+\frac{2\mathcal{R}_{\max}D_{\mathrm{TV}}[p_\pi^{\hat{M}}(\cdot|s),p_\pi^M(\cdot|s)]}{(1-\gamma)^2}\\&\le\mathcal{J}_{\hat{M}}(\pi_{\hat{M}})+\frac{2\mathcal{R}_{\max}D_{\mathrm{TV}}[p_\pi^{\hat{M}}(\cdot|s),p_\pi^M(\cdot|s)]}{(1-\gamma)^2}\\&\le\mathcal{J}_M(\pi_{\hat{M}})+\frac{4\mathcal{R}_{\max}D_{\mathrm{TV}}[p_\pi^{\hat{M}}(\cdot|s),p_\pi^M(\cdot|s)]}{(1-\gamma)^2}\end{aligned} \tag{8.48}$$

With the help of expectation over the distribution of MDPs, the final equation of a lower bound can be drawn as that in **Theorem** (2).

$$\mathbb{E}_{M\sim p(M)}\Big[\mathcal{J}_M(\pi_{\hat{M}})\Big]\ge\mathbb{E}_{M\sim p(M)}\Big[\mathcal{J}_M(\pi_M)\Big]-\frac{4\epsilon\mathcal{R}_{\max}}{(1-\gamma)^2} \tag{8.49}$$

C.4 *GSSM Modules in PyTorch*

Our work GSSM is built upon structures of GNNs (Kipf and Welling, 2016; Satorras and Estrach, 2018; Wang et al., 2018). However, unlike vanilla GNNs, we try to learn the normalized graph Laplacian in modeling. Also, GSSMs make use of message passing in GNN modules and transform the learned node representations into target transition latent variable z_t and the prior global latent variable z_c for an MDP.

Here we rewrite the graph convolutional operation in the form of a node feature matrix $\mathcal{V}_C$, which is easier to implement in programming. The graph convolution operator $\circ$ over any data point x_t can be defined,

$$f(\mathcal{V}_C) = \sigma\Big(D^{-1}L[\mathcal{V}_C\mathcal{W}]\Big) \tag{8.50a}$$

$$\mathcal{G}(x_t) \circ f = \sum_{i \in O_t} f(v_i)\texttt{sim}(x_t, x_i) \tag{8.50b}$$

where $\mathcal{V}_C$ is the feature matrix of the context points, and $\mathcal{W}$ denotes a trainable layer matrix. The weight parameter is $\texttt{sim}(x_t, x_i)$ and $f(v_i)$ is the embedding of a node in the graph $\mathcal{G}$ after message passing processes from its neighbors O_t. The graph Laplacian matrix L reveals the connection relationship, where D is a diagonal matrix to normalize the row elements in Eq. (8.50). These correspond to the left side of Fig. (16). The equivalent element-wise graph operation can be found in Eq. (5.4). We refer the reader to our GitHub code for more information about graph convolution operations and the induced meta dynamics model.

C.5 *Evidence Lower Bound for GSSM*

Here $P(D)$ denotes the distribution of state action pairs in meta-training processes, and each data point is attached with a context set $[x_c, y_c]$ (a batch of transition data points) to imply the statistics information from a task. With a Jessen's inequality and an approximate posterior $q_\phi(z_t|x_t, x_c, y_c)$, we can have evidence lower bound as follows.

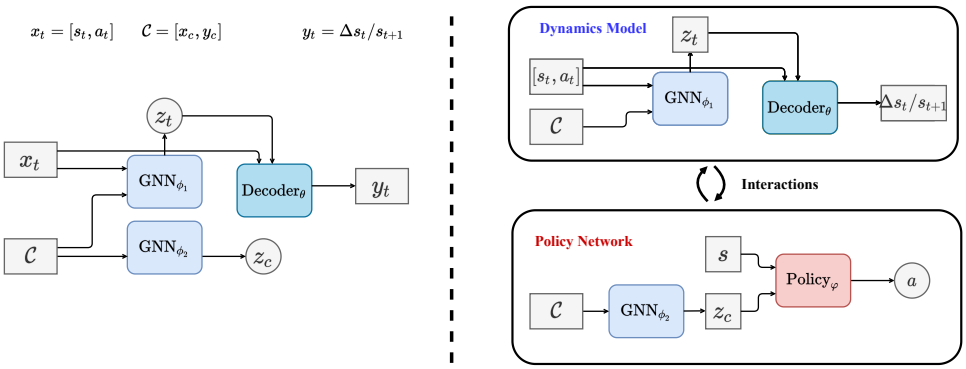
$$\begin{aligned}\mathbb{E}_{p(D)} \ln p(y_t|x_t, x_c, y_c) &= \mathbb{E}_{p(D)} \ln \mathbb{E}_{q_\phi}\Big[\frac{p(z_*)}{q_\phi(z_t|x_t, x_c, y_c)} p_\theta(y_t|x_t, z_t)\Big] \\ &\geq \mathbb{E}_{p(D)}\left[\mathbb{E}_{q_\phi} \ln\Big[p_\theta(y_t|x_t, z_t)\Big] - \mathbb{E}_{q_\phi} \ln\Big[\frac{q_\phi(z_t|x_t, x_c, y_c)}{p(z_t)}\Big]\right]\end{aligned} \tag{8.51}$$

By replacing the zero information prior distribution $p(z_t)$ with $q(z_t|x_c, y_c)$, we can derive the formerly mentioned ELBO.

$$\mathbb{E}_{p(D)}\Big[\ln \underbrace{p(y_t|x_t,x_c,y_c)}_{\text{intractable data likelihood}}\Big] \geq \mathbb{E}_{p(D)}\Big[\mathbb{E}_{q_{\phi_1}}[\ln p_\theta(y_t|x_t,z_t)] - D_{KL}[\underbrace{q_{\phi_1}(z_t|x_t,x_c,y_c)}_{\text{approximate posterior}} \parallel \underbrace{q_{\phi_2}(z_c|x_c,y_c)}_{\text{approximate prior}}]\Big] \tag{8.52}$$

Note that both the approximate prior and the posterior are learnable with a partially shared neural network in meta learning scenarios, which is similar in work (Denton and Fergus, 2018; Pertsch et al., 2020; Garnelo et al., 2018b). And the learned prior $q_{\phi_2}(z_c|x_c,y_c)$ can be viewed as a summary of context points, which is further used to help induce amortized policies $\pi_\varphi(a|s,z_c)$. For more details on the encoding relationship between these context points and target points, refer to Fig. (16).

Besides, the number of context points is random smaller than the batch size in meta training dynamics models, which shares the same setting as Neural Processes. As mentioned in the Main Paper, a fully connected graph is built among context points, and the graph Laplacian matrix is learned based on Eq. (5.4.a). Here we treat all the context points as the neighborhood $\mathcal{O}_t$ of a target point x_t. The way of constructing fully connected graphs is a limitation for GSSMs, and future work can be the discovery of optimal graph structures to improve performance further.



(a) Computational Graph in Meta-training Dynamics Models

(b) Meta-training Amortized Policies

Figure 36: Computational Graphs of GSSM in Meta-training Processes. Note that amortized policy search is used here. On the **Left** side of the Figure: it describes the connections of variables in neural networks for meta dynamics models. On the **Right** side of the Figure: it illustrates the process in meta model-based policy search (Note that the reparameterization trick is used here and sampled latent variables z_t and z_c are deterministic, denoted by squares).

C.6 *Computational Graphs and Detailed Descriptions*

Here we add more explanations about our developed GSSM and amortized policy search. This section corresponds to Fig. (16). Especially, computational processes are displayed in Fig. (36).

On the **Left** of Fig. (36), the $\texttt{GNN}_{\phi_2}$ module is used to summarize the task using $q_{\phi_2}(z_c)$ from the context points C, while the $\texttt{GNN}_{\phi_1}$ module is used to learn state-action pair x_t dependent latent variables $q_{\phi_1}(z_t)$. As mentioned in the Main paper, these two modules share part of the parameters. The concatenation of x_t and sampled z_t is input into the $\texttt{Decoder}_\theta$ module to predict the transited state s_{t+1} or state difference Δs_t.

On the **Right** of Fig. (36), this depicts the process in learning amortized policies (Mainly refer to the loop iterations `step (6)-(10)` in Algorithm (6)). Given a batch of MDPs, context points C are sampled using a uniform random policy and input into $\texttt{GNN}_{\phi_2}$ to formulate task-specific latent variables. Note that the reparameterization trick is used in this process, which means we sample $\epsilon \sim \mathcal{N}(0, I)$ during the loop of iterations to formulate $z_t = \mu_t + \Sigma_t^{-\frac{1}{2}}\epsilon$ for state-action pairs $x_t = [s_t, a_t]$. This operation can be interpreted as sampling MDPs from the posterior in posterior sampling (Osband et al., 2013). Similarly, we sample $\epsilon \sim \mathcal{N}(0, I)$ to formulate $z_c = \mu_c + \Sigma_c^{-\frac{1}{2}}\epsilon$ and get this sample value retained in the loop iterations (In Actor-Critic cases, the sampled value also takes part in value function approximators). During the process, these task-specific policies interact with a collection of sampled MDPs to seek optimal results like that in BOSS (Asmuth et al., 2009), which corresponds to finding optimal policies w.r.t sampled MDPs in posterior sampling. In the next loop of iterations, new transitions are collected using the amortized policy, and the dynamics model is retrained to update the posterior.

C.7 *Policy Gradient Estimates in Amortized Policy Search*

Due to page limits, we provide more details in actor critic cases. As revealed in Fig. (16), the right one is to show amortized policy search in a collection of approximate dynamics models. Estimates of policy gradients are formulated, and these details will tell readers how our amortized policies are learned in MBMRL. All of these correspond to `step (10)` in Algorithm (6).

Note that parameters φ in our amortized policies $\pi_\varphi(a|[s, z_c])$ are optimized in developed dynamics models. Dynamics models as GSSMs consist of parameters θ and $\phi = [\phi_1, \phi_2]$, and we denote a sampled approximate dynamics model for one task as $\hat{\mathcal{M}}$. As for a distribution of synthetic trajectories τ from a learned dynamics model $\hat{\mathcal{M}}$, we use $p(\tau|\hat{\mathcal{M}}; \varphi, \phi)$ to define, where q_{ϕ_2} encodes contextual information from $[x_c, y_c]$ for different tasks. Given a sampled trajectory τ from an approximate dynamics model $\hat{\mathcal{M}}$, we can decompose it according to the Markov property.

$$p(\tau|\hat{\mathcal{M}}; \varphi, \phi) = p(s_0) \prod_{t=0}^{T-1} [p(s_{t+1}|s_t, a_t, z_t)\pi_\varphi(a_t|[s_t, z_c])] \tag{8.53}$$

Besides, the proposed amortized policy search strategy can be combined with any other dynamics model with contextual latent variables, not limited to Graph Structured Surrogate Models in our paper (*e.g.*, NP (Garnelo et al., 2018b; Galashov et al., 2019), it is also available to combine with our developed policy search strategy).

We consider *actor-critic* frameworks, where a value function is also conditioned on the latent variable (Refer to the right side of Fig. (16). Hence, two objectives, namely value function approximation and policy optimization, are involved in this setting.

$$\mathcal{L}_{\mathrm{C}}(\hat{\varphi}) = \mathbb{E}_{\hat{\mathcal{M}}\sim p(\hat{\mathcal{M}};\theta,\phi)} \mathbb{E}_{\substack{(s,a,s',r)\sim\mathcal{B}(\hat{\mathcal{M}}) \\ z_c\sim q_\phi(z_c|[x_c,y_c])}} \left[Q_{\hat{\varphi}}([s,z_c],a) - (r+\gamma V_{\tilde{\varphi}}([s',z_c]))\right]^2 \tag{8.54}$$

The value function approximation objective is Eq. (8.54), where $\mathcal{B}(\hat{\mathcal{M}})$ is a batch of synthetic transition samples from $\hat{\mathcal{M}}$ and $\hat{\varphi}$ is the parameter of value function approximators. The critic optimization process is inside policy search during meta-training processes via gradient updates. The actor optimization objective can be found in the main paper.

To optimize policy functions as that in Eq. (5.12), we can utilize a likelihood ratio trick (Williams, 1992) and compute the derivative in transitions as follows.

$$\ln p(\tau|\hat{\mathcal{M}};\varphi,\phi) = \ln p(s_0) + \sum_{t=0}^{T-1} \left[\ln p(s_{t+1}|s_t,a_t,z_t)) + \ln \pi_\varphi(a_t|[s_t,z_c])\right] \tag{8.55}$$

$$\nabla_\varphi \ln p(\tau|\hat{\mathcal{M}};\varphi,\phi) = \nabla_\varphi \ln \cancel{p(s_0)} + \sum_{t=0}^{T-1} \nabla_\varphi \left[\ln \cancel{p(s_{t+1}|s_t,a_t,z_t)} + \ln \pi_\varphi(a_t|[s_t,z_c])\right] \tag{8.56}$$

Based on Eq. (8.56), we formulate the estimated policy gradient in Eq. (8.57), and a modified PPO (Schulman et al., 2017) is used as an instantiation to implement in our settings.

$$\begin{aligned} \nabla_\varphi \mathcal{J}(\varphi) &= \mathbb{E}_{\hat{\mathcal{M}}\sim p(\mathcal{M};\theta,\phi)} \left[\int \nabla_\varphi p(\tau|\hat{\mathcal{M}};\varphi,\phi)\mathcal{R}(\tau)d\tau\right] \\ &= \mathbb{E}_{\hat{\mathcal{M}}\sim p(\mathcal{M};\theta,\phi)} \mathbb{E}_{\tau\sim p(\tau|\hat{\mathcal{M}};\varphi,\phi)} \left[\nabla_\varphi \ln p(\tau|\hat{\mathcal{M}};\varphi,\phi)\mathcal{R}(\tau)\right] \\ &= \mathbb{E}_{\hat{\mathcal{M}}\sim p(\mathcal{M};\theta,\phi)} \mathbb{E}_{\tau\sim p(\tau|\hat{\mathcal{M}};\varphi,\phi)} \left[\sum_{t=0}^{T-1} \nabla_\varphi \ln \pi(a_t|[s_t,z_c])\right] \cdot \left[\sum_{t=0}^{T-1} r(s_t,a_t,s_{t+1})\right] \end{aligned} \tag{8.57}$$

Meanwhile, the policy gradient in the form of advantage functions is derived in Eq. (8.58),

$$\nabla_\varphi \mathcal{J}(\varphi) = \mathbb{E}_{\substack{\hat{\mathcal{M}}\sim p(\hat{\mathcal{M}};\theta,\phi) \\ \tau\sim p(\tau|\hat{\mathcal{M}};\varphi,\phi)}} \left[\sum_{t=0}^{T-1} \nabla_\varphi \ln \pi(a_t|[s_t,z_c]) \cdot A_t([s_t,z_c],a_t)\right] \tag{8.58}$$

where $A_t([s_t,z_c],a_t)$ is an advantage function, mostly written as the difference between a cumulative reward term and a baseline term $A_t([s_t,z_c],a_t) = \sum_{t'=t+1}^{T-1} r([s_{t'},z_c],a_{t'}) - b_t([s_t,z_c])$.

Similar to the term on the right side of Eq. (5.12), the corresponding Monte Carlo estimate of the policy gradient can be easily obtained from Eq. (8.58), and we skip this step in this section.

C.8 *Experimental Settings and Training Details*

In this section, we provide information about environments and give more details in meta training/testing processes. Especially, the updated code file can be found in (`https://github.com/hhq123gogogo/GSSM_APS`). Only one layer graph convolution is used in general settings.

Environmental Details

MBMRL Tasks. Here we describe meta reinforcement learning tasks in this paper. The Cart-Pole environment can be found in the link[12] here. The Acrobot is based on open-ai gym[13]: with continuous states $[\theta_1, \theta_1', \theta_2, \theta_2']$ as angles and instant angle velocities, the goal is to sequentially select an action from $\{-1, 0, +1\}$ (respectively Right Torque, No Torque, Left Torque) to reach the height above the top of the pendulum as early as possible. Half-Cheetah/Slim-Humanoid are from a Mujoco package[14] and both are to conduct locomotion tasks. Generations of diverse Cart-Pole/Acrobot environments have been introduced in the main paper. As for Cart-Pole tasks, 50 unseen tasks are sampled from simulators for meta testing, and each task is with 50 episodes in evaluation (refer to Fig. (17)). As for Acrobot tasks, 33 unseen tasks are sampled from simulators for meta testing and each task is with 50 episodes in evaluation (refer to Table (9)/(25)). As for configurations of Half-Cheetah/Slim-Humanoid environments, we generate the Meta-training MDPs via the combination of the mass re-scaled coefficient in the list $\{0.8, 0.9, 1.0, 1.1, 1.2\}$ and the damping coefficient in the list $\{0.8, 0.9, 1.0, 1.1, 1.2\}$, while those hyper-parameters for Meta-testing phases are $\{0.85, 0.95, 1.05, 1.15\}$ for both mass-rescaled and damping coefficients. As a result, totally 16 unseen MDPs are generated by the Cartesian of mass coefficients and damping coefficients for meta-testing processes (refer to Table (25) and Table (9)/(25)).

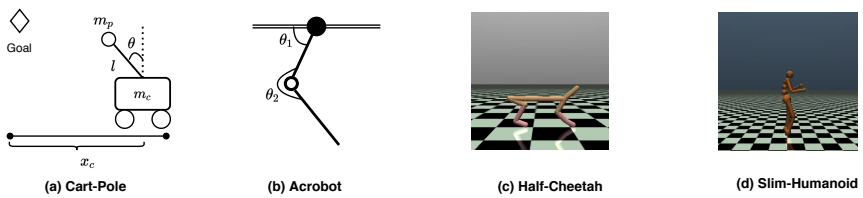


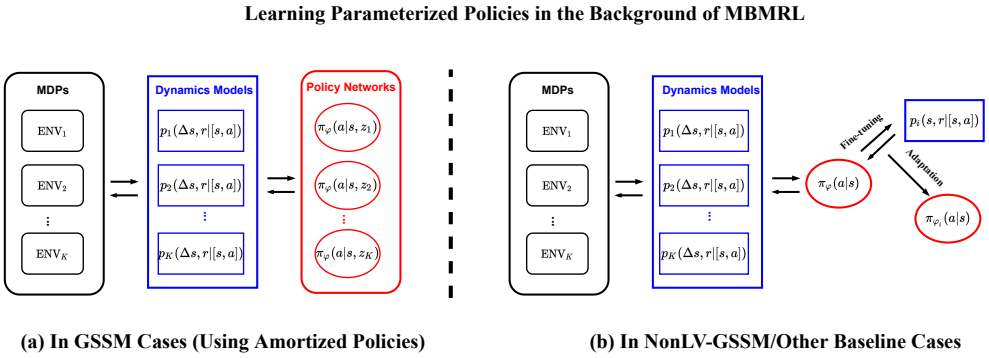
Figure 37: Fundamental Environments used in Meta Model-based Reinforcement Learning Experiments.

12 https://github.com/BrunoKM/deep-pilco-torch
13 https://gym.openai.com/
14 http://www.mujoco.org/

Table 26: Reward Functions in Related Environments.

Env	Reward Functions	Horizon	Control
Cart-Pole	$1 - \exp\left\{\left(-\frac{\|d^2\|}{\sigma_c^2}\right)\right\}$	25	Continuous
Acrobot	$\text{bool}(-l_1 \cos(\theta_1) - l_2 \cos(\theta_1 + \theta_2) - l_1)$	200	Discrete
H-Cheetah	$\frac{x_{t+1} - x_t}{\nabla t} - 0.1 * \|\|a_t\|\|_2^2$	1000	Continuous
S-Humanoid	$\frac{50(x_{t+1} - x_t)}{3\nabla t} - 0.1 * \|\|a_t\|\|_2^2 + 5.0 * \text{bool}(1.0 \leq x_{t,h} \leq 2.0)$	1000	Continuous

Reward Descriptions. Besides, reward functions are listed here (refer to Table (26)). More details are as follows. In Cart-Pole environments, d in a reward function measures the square of the distance between the pole's endpoint and its goal, and hyper-parameter $\sigma_c = 0.25$. In Acrobot environments, the list of parameters $\{l_1, l_2, \theta_1, \theta_2\}$ corresponds to Fig. (37) in terms of meanings in a reward function. In Half-Cheetah environments, x_t is the notation of the x-coordinate in the Half-Cheetah agent at time slot index t, ∇_t is the time difference in dynamics (the resulted ratio is the speed of agent.), and a_t is the action performed instantly. In Slim-Humanoid environments, notations are similar to those in Half-Cheetah and $x_{t,h}$ in reward functions refer to the instant torso's height. Horizons of trajectories, as well as types of action spaces, can also be found in Table (26).



(a) In GSSM Cases (Using Amortized Policies) (b) In NonLV-GSSM/Other Baseline Cases

Figure 38: Meta Model-based Policy Search used in Models. Note that reducing adaptation time in policies is the first priority in this work. On the **Left**, amortized policies are used and latent variables are to specify different tasks. On the **Right**, the meta-trained policy is not conditioned on latent variables and needs to be adapted to respective tasks, which consumes additional time.

Training Details

Data Preprocessing. In Acrobot tasks, the output of dynamics models is the next or transited state ($x = [s, a]$, $y = s'$). In other tasks, the output of dynamics models is the difference between the next state and the current state ($x = [s, a]$, $y = \Delta s$). For the input of dynamics models, it is the state-action pair in all environments. For Half-

Cheetah/Slim-Humanoid environments, standardization is required for both the input and the output of dynamics models in meta training processes.

More Details in Policy Search. In Cart-Pole Swing-Up environments, Back-Propagation Through Time (BPTT) is used in model-based policy search and the policy network parametrized with a radial basis function follows that in (Gal et al., 2016), for GSSM+APS, a latent variable is concatenated with the state variable as the input. In Acrobot/Half-Cheetah/Slim-Humanoid, we combine PPO with the learned dynamics model (We also perform additional trials in BPTT strategies but this kind of model-based policy search suffers from gradient exploding in practice), and a direct combination of model-based and model-free RL algorithms in meta-learning leads to stable training.

Besides, meta-trained policies in MLSM-v0/MLSM-v1/M-DPILCO require additional policy gradient updates in separate dynamics models of tasks, and these are up to tasks based on our trials. For Cart-Pole/Acrobot, five trajectories are enough to fine-tune these policies, and for Half-Cheetah/Slim-Humanoid one trajectory is enough to fine-tune these policies. Such adaptation in MLSM-v0, MLSM-v1, and M-DPILCO consumes additional time when the learned policy is implemented in unseen environments.

The pipeline is reflected in Fig. (38). Traditional model predictive control strategies are prohibitively expensive in implementation, costing much more time with lower efficiency in high dimensional action space. Since related work employing parameterized policies in MBMRL remains limited, our proposed method can be deemed as a preliminary exploration.

Further Descriptions in Fig.s/Tables. Here we add more descriptions on Cart-Pole, where authors can follow the implementations in the work[15], and the state-of-art performance using Deep-PILCO is about -0.6 in episodes for a single task. We also try DR-PPO and PE-PPO in Cart-Pole tasks with more than 10x required time steps in training, but the resulted performance in testing is far worse than MBMRL ones, and we guess the PPO algorithm here cannot well handle planning with short horizons (Other referred model-free results can be found in (Lillicrap et al., 2016). In addition, the estimated required samples of all MBMRL baselines for Cart-Pole are even 2x less than model-free ones (Lillicrap et al., 2016; Gal et al., 2016) to train in one single MDP.).

In Fig. (17.(a)/(b)/(d)), each `iter` in x-axis indicates that a new trajectory is sampled to update the dynamics buffer, the batch size of samples in dynamics memory buffer is 100, the default epoch in training dynamics models is 5 in each `iter`, and the Cart-Pole environment changes every every 10 `iter`. Meanwhile, for each `iter`, 25 trajectories are sampled using the updated policy to average results for evaluating the performance in the trained task (this process results in learning curves).

Fig. (19) keeps track of meta-training performance using MBMRL algorithms in Acrobot/H-Cheetah/S-Humanoid, and dynamics of MDPs change with iterations (the batch numbers of tasks are 1 for Acrobot/H-Cheetah and 3 for S-Humanoid). In Acrobot, the batch size of samples in the dynamics memory buffer is 100, the default epoch in training dynamics models is 20 in each `iter`, and 15 trajectories are sampled for evaluation to show performance on learning curves. In H-Cheetah, the batch size of

15 https://github.com/BrunoKM/deep-pilco-torch

samples in the dynamics memory buffer is 1000, the default epoch in training dynamics models is 20 in each `iter`, and 15 trajectories are sampled for evaluation to show performance on learning curves. In S-Humanoid, the batch size of samples in dynamics memory buffer is 1000, the default epoch in training dynamics models is 10 in each `iter`, and 15 trajectories are sampled for evaluation to show performance on learning curves. And every fixed number of iterations, MDPs in meta-training are resampled (for Acrobot, every 3 `iters`; for H-Cheetah, every 3 `iters`; for S-Humanoid, every 1 `iter`).

Table (25) and Table (8)/(9) summarize the meta-testing results over unseen MDPs. We collect the rewards in each task using these models to obtain the average results in each task and then report average rewards over meta testing tasks in the table. Some additional explanations are as follows. In meta-testing tasks of Cart-Pole, contextual latent variables in GSSM/MLSM-v0/MLSM-v1 are computed after transitions of two trajectories (50 transition steps) are aggregated. In meta-testing tasks of Acrobot, contextual latent variables in GSSM/MLSM-v0/MLSM-v1 are computed after transitions of a one-sixth trajectory (50 transition steps) are aggregated. In meta-testing tasks of Half-Cheetah/Slim-Humanoid, contextual latent variables in GSSM/MLSM-v0/MLSM-v1 are computed after transitions of a half trajectory (500 transition steps) are aggregated.

Meanwhile, meta-training processes in model-free meta reinforcement learning are recorded in Fig. (19). All these are trained with Adam optimizers, and learning rates are 5e-4 in default.

C.9 *Neural Architectures and Parameter Settings*

Here neural architectures in meta dynamics models are listed in Table (27). These architectures are shared across all implemented tasks in the paper. And one layer graph encoding is enough to guarantee performance in our GSSM implementations for all experiments. For Meta-DPILCO, neural architectures resemble that in the table except that encoders for latent variables are removed, and dropout modules are integrated in each layer. In Cart-Pole environments, parameters in Table (27) are $\{n = 2, dim\_latxy = 32, dim\_lat = 16, m = 2, dim\_h = 200\}$. In Acrobot environments, parameters in Appendix Table (27) are $\{n = 2, dim\_latxy = 32, dim\_lat = 16, m = 5, dim\_h = 400\}$. In Mujoco environments, parameters in Appendix Table (27) are $\{n = 2, dim\_latxy = 32, dim\_lat = 16, m = 5, dim\_h = 400\}$. Also, note that in our implementations, we set $dim\_lat = 8$ for GSSM in Cart-Pole/Acrobot/H-Cheetah because lower dimensional information bottlenecks are more compact and help amortized policy search, while for models using non-latent variable conditioned policies better results are achieved with information bottleneck $dim\_lat = 16$.

As for meta policy networks or latent variable conditioned policy networks (used in GSSM), we adopt the ordinary ones, and these are listed in Table (28). In Cart-Pole environments, parameters in Table (28) are $\{n_p = 1, dim\_ph = 50\}$. In Acrobot environments, parameters in Appendix Table (28) are $\{n_{pa} = 1, dim\_ph = 128, n_{pc} = 1\}$. In Half-Cheetah environments, parameters in Table (28) are $\{n_{pa} = 1, dim\_ph = 128, n_{pc} = 1\}$. In model-free meta reinforcement learning scenarios, the contextual encoder is permutation invariant, the same as that used in MLSM-v0, and the optimization

Table 27: Neural Network Structure of MBMRL Models. The transformations in the table are linear, followed with ReLU activation mostly. As for MLSM-v1, the encoder network is doubled in the table since there exists a local variable for prediction.

NP Models	Encoder	Decoder
MLSM-v0/v1	$[dim\_x, dim\_y] \mapsto \underbrace{dim\_latxy \mapsto dim\_latxy}_{n \text{ times}}$ $dim\_latxy \mapsto dim\_lat.$	$[dim\_x, (2*)dim\_lat] \mapsto \underbrace{dim\_h \mapsto dim\_h}_{m \text{ times}}$ $dim\_h \mapsto dim\_y$
GSSM	$[dim\_x, dim\_y] \mapsto \underbrace{dim\_latxy \mapsto dim\_latxy}_{n \text{ times}}$ $dim\_x \mapsto dim\_latx;$ $[dim\_latx, dim\_laty] \mapsto dim\_lat.$	$[dim\_x, dim\_lat] \mapsto \underbrace{dim\_h \mapsto dim\_h}_{m \text{ times}}$ $dim\_h \mapsto dim\_y$

Table 28: Neural Network Structure in Meta Policy Networks. For Back-propagation Through Time (BPTT) and Actor-Critic Policy Gradient Algorithms, neural architectures are different. ReLU is used as an activation function. Soft-max is used in the output of Actor Networks in the discrete control.

Policy Search	Neural Architectures
BPTT	$[dim\_obs] \,/\, [dim\_obs, dim\_lat] \mapsto \underbrace{dim\_ph \mapsto dim\_ph}_{n_p \text{ times}}$ $dim\_ph \mapsto dim\_act.$
AC-PG (PPO)	$[dim\_obs] \,/\, [dim\_obs, dim\_lat] \mapsto \underbrace{dim\_ph \mapsto dim\_ph}_{n_{pc} \text{ times}} \mapsto dim\_act$ (Actor Network) $[dim\_obs] \,/\, [dim\_obs, dim\_lat] \mapsto \underbrace{dim\_ph \mapsto dim\_ph}_{n_{pa} \text{ times}} \mapsto 1$ (Critic Network).

objectives follow those in PEARL (Rakelly et al., 2019). Besides, we implement the vanilla version of PEARL with the same training sample volume as that in MBMRL but find the results are inferior to mentioned model-free baselines.

D SUPPLEMENTARY MATERIALS IN CHAPTER 6

D.1 *Probabilistic Generative Process in NPs*

Here we can translate the generative process of NPs in the following mathematical way.

$$\begin{aligned} \tau \sim p(\mathcal{T}), \quad z \sim \mathcal{N}(z; \mu_\vartheta(\mathcal{D}_\tau^C), \Sigma_\vartheta(\mathcal{D}_\tau^C)) \\ x_i \sim p(x), \quad y_i \sim p(y|[x_i, z]; \vartheta) \quad \forall i \in \{1, 2, \ldots, n+m\} \end{aligned} \tag{8.59}$$

D.2 *Run-time Complexity of Predictive Distributions in GPs & NPs*

Take one-dimensional deep Gaussian processes (Dai et al., 2016b) as an example. With context points $\mathcal{D}^C = \{(x_i, y_i)\}_{i=1}^n$ and target points $\mathcal{D}^T = \{(x_i, y_i)\}_{i=1}^{n+m}$, the key to applications is the predictive distribution $p(f(x_T)|\mathcal{D}^C, x_T) = \mathcal{N}(y_T; \mu_T, \Sigma_T)$. The conditional mean μ_T and covariance Σ_T functions in Eq. (8.60) are permutation invariant to the order of context points.

$$\begin{aligned} \mu_T &= m_\theta(x_T) + \Sigma_{T,C}\Sigma_{C,C}^{-1}\big(y_C - m_\theta(x_C)\big) \\ \Sigma_T &= \Sigma_{T,T} - \Sigma_{T,C}\Sigma_{C,C}^{-1}\Sigma_{C,T} \end{aligned} \tag{8.60}$$

Here the covariance matrix denoted by Σ is computed with the context input $x_C = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$, the target input $x_T = (x_1, \ldots, x_{n+m}) \in \mathbb{R}^{(n+m) \times d}$, and a kernel function ψ, *e.g.* $[\Sigma_{C,C}]_{i,j} = \psi(x_i, x_j)$, m_θ is the mean function m_θ, and the context output is $y_C = (y_1, \ldots, y_n) \in \mathbb{R}^n$. Nevertheless, the computation of matrix inversion in Eq. (8.60) makes the runtime complexity as expensive as $\mathcal{O}((n+m)^3)$. As comparison, the runtime complexity in NPs is $\mathcal{O}(n+m)$ since NPs enable predictions by running the stochastic forward pass with one Monte Carlo sample.

D.3 *NPs Formulation & Structural Inductive Biases*

Prior, Posterior & Proposal Distributions

Since fast adaptation is achieved in an amortized way, which reduces the gradient updates *w.r.t.* model parameters to learning function specific latent variables with amortized networks. The context points are treated as a set and the amortized network should be permutation invariant *w.r.t.* the order of data points.

- **Approximate Posterior Distribution.** This is denoted by $q_\phi(z|\mathcal{D}_\tau^T)$ in this chapter. Usually, the approximate posterior is used in NPs (Garnelo et al., 2018b; Kim et al., 2019a) and works as a proxy for the non-analytical real posterior $p(z|\mathcal{D}_\tau^T; \vartheta)$.
- **Prior Distribution.** This is denoted by $p(z|\mathcal{D}_\tau^C; \vartheta)$ in this chapter. Unlike the approximate prior $q_\phi(z|\mathcal{D}_\tau^C)$ used in NPs, we use an exact functional prior in SI-NPs.
- **Proposal Distribution.** This is denoted by $q_\eta(z|\mathcal{D}_\tau^T)$ in this chapter. The role of the proposal distribution resembles that of the approximate posterior in NPs. It

is used to sample latent variables and enable the computation of the importance weights in NPs.

Since the latent variable z is inferred from a set of data points, the distributions $q_\phi(z)$ and $q_\phi(z|\mathcal{D}_\tau^C)$ should be permutation invariant to the order of data points. An example module to parameterize $q_\phi(z|\mathcal{D}_\tau^C) = \mathcal{N}(z;\mu_\tau,\Sigma_\tau)$ can be Eq. (8.61) with $\bigoplus$ a mean pooling operator and $\{h_\phi, g_\phi\}$ encoder networks.

$$r_i = h_\phi([x_i,y_i])\ \forall (x_i,y_i)\in\mathcal{D}_\tau^C,\quad r_C = \bigoplus_{i=1}^{N} r_i,\quad [\mu_\tau,\Sigma_\tau] = g_\phi(r_C) \tag{8.61}$$

The same with that in traditional stochastic processes, a realisation corresponds to a sampled function $f(X)$ generated in a sequential way: $z\sim q_\phi(z|\mathcal{D}_\tau^C)$, $f(X)\sim p(Y|X,z;\vartheta)$.

Approximate ELBOs in NPs

$$\ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta) = \ln\int p(\mathcal{D}_\tau^T|z;\vartheta)p(z|\mathcal{D}_\tau^C;\vartheta)dz \tag{8.62a}$$

$$\geq \mathbb{E}_{q_\phi(z|\mathcal{D}_\tau^T)}\Big[\ln p(\mathcal{D}_\tau^T|z;\vartheta)\Big] - D_{KL}\Bigg[\underbrace{q_\phi(z|\mathcal{D}_\tau^T)}_{\text{Approximate Posterior}}\ \Big\|\ \underbrace{p(z|\mathcal{D}_\tau^C;\vartheta)}_{\text{Intractable Prior}}\Bigg] = \mathcal{L}_{\text{ELBO}}(\vartheta,\phi) \tag{8.62b}$$

$$\approx \mathbb{E}_{q_\phi(z|\mathcal{D}_\tau^T)}\Big[\ln p(\mathcal{D}_\tau^T|z;\vartheta)\Big] - D_{KL}\Bigg[\underbrace{q_\phi(z|\mathcal{D}_\tau^T;\vartheta)}_{\text{Approximate Posterior}}\ \Big\|\ \underbrace{q_\phi(z|\mathcal{D}_\tau^C)}_{\text{Approximate Prior}}\Bigg] = \mathcal{L}_{\text{NP}}(\vartheta,\phi)\quad\square \tag{8.62c}$$

For Eq. (8.62.b), remember that *w.r.t.* these VAE-like methods, there is no improvement guarantee of the evidence in each iteration when optimizing ELBO due to the existence of posterior approximation gap. Importantly, the form of the functional prior is unknown.

Vanilla NPs directly replace the real functional prior by the approximate one $q_\phi(z|\mathcal{D}_\tau^C)$ and introduce the consistent regularizer in Eq. (6.5). We further introduce the prior approximation gap in Eq. (8.63), in which the sign is undetermined.

$$\mathcal{L}_{\text{NP}}(\vartheta,\phi) = \mathcal{L}_{\text{ELBO}}(\vartheta,\phi) + \underbrace{\mathbb{E}_{q_\phi(z)}\left[\ln\frac{q_\phi(z|\mathcal{D}_\tau^C)}{p(z|\mathcal{D}_\tau^C;\vartheta)}\right]}_{\text{Prior Approximation Gap}} \tag{8.63}$$

Based on decomposition in Eq. (6.4)/(8.63), we can find there exists no consistent monotonic relationship between $\mathcal{L}(\vartheta)$ and $\mathcal{L}_{\text{NP}}(\vartheta, \phi)$. The invalid ELBO makes the optimization *w.r.t.* $\mathcal{L}_{\text{NP}}(\vartheta, \phi)$ not always improve log-likelihood $\mathcal{L}(\vartheta)$.

$$\mathcal{L}(\vartheta) \geq \mathcal{L}_{\text{ELBO}}(\vartheta, \phi), \quad \mathcal{L}(\vartheta) \ngeq \mathcal{L}_{\text{NP}}(\vartheta, \phi) \quad \forall \vartheta \in \Theta \text{ and } \phi \in \Phi \tag{8.64}$$

Also, the right side of Eq. (8.64) indicates the previous commonly-used strategies, *e.g.* normalizing flows for richer variational posterior distributions (Rezende and Mohamed, 2015), auxiliary variables for augmented variational posterior distributions (Maaløe et al., 2016) and more flexible prior distributions (Tomczak and Welling, 2018), to close VAEs inference gaps (Cremer et al., 2018) will not guarantee the performance improvement in a theoretical sense. In other words, the consistent regularizer in NPs is problematic for optimization.

Inference Gaps

In this section, we apply the trick of inference gap decomposition (Cremer et al., 2018) to understand vanilla NPs. Here we denote the approximate inference gap by D_{KL}^{AI} and the posterior approximation gap by D_{KL}^{PA}.

Table 29: **Inference Gaps in vanilla NPs.** The $\downarrow$ indicates the minimization to obtain the optimal inference solution. The sign $--$ means not applicable in deriving equivalent KL divergence form. The sign * indicates the optimal posterior approximation in the family of variational distributions $\phi^* = \arg\min_{\phi \in \Phi} D_{KL}\Big[q_\phi(z) \parallel p(z|\mathcal{D}_\tau^T; \vartheta^*)\Big]$. Here ϑ^* consists of the optimal parameters in priors $p(z|\mathcal{D}_\tau^C; \vartheta^*)$ and the conditional distribution $p(\mathcal{D}_\tau^T|z; \vartheta^*)$.

Terms	Optimization Objective	KL Divergence or Gaps
Approximate Inference	$\mathcal{L}(\vartheta^*) - \mathcal{L}_{\text{ELBO}}(\vartheta^*, \phi) \downarrow$	$D_{KL}\Big[q_\phi(z) \parallel p(z^*|\mathcal{D}_\tau^T; \vartheta)\Big]$
Posterior Approximation	$\mathcal{L}(\vartheta^*) - \mathcal{L}_{\text{ELBO}}(\vartheta^*, \phi^*) \downarrow$	$D_{KL}\Big[q_{\phi^*}(z) \parallel p(z|\mathcal{D}_\tau^T; \vartheta^*)\Big]$
Amortization	$\mathcal{L}_{\text{ELBO}}(\vartheta^*, \phi^*) - \mathcal{L}_{\text{ELBO}}(\vartheta^*, \phi) \downarrow$	$D_{KL}^{\text{AI}} - D_{KL}^{\text{PA}}$
NP Prior Approximation	$\lvert\mathcal{L}_{\text{ELBO}}(\vartheta^*, \phi^*) - \mathcal{L}_{\text{NP}}(\vartheta^*, \phi^*)\rvert$	–
Surrogate Likelihood	$\mathcal{L}(\vartheta^*) - \mathcal{L}(\vartheta; \vartheta_k) \downarrow$	$\mathcal{L}(\vartheta^*) - \mathcal{L}(\vartheta_H)$

D.4 *Neural Architectures in Implementations*

D.5 *Formulation of Variational Expectation Maximization Method*

In this section, we at first present the optimization objective in VAE-like methods. The concept of meta learning surrogate functions is introduced and NPs are verified. Meanwhile, the improvement guarantee as well as other concerning technical points are included to better understand our method.

Proof of Meta Learning Surrogate Functions

The proof is directly based on **Definition** (3). When an objective function $f(\vartheta)$ to maximize is complicated, *e.g.* multi-modal likelihood functions, a surrogate function $g(\vartheta;\vartheta_k)$ enables an easier proxy implementation with convergence guarantee to at least the local optimal. To see this point, recall the properties of the surrogate function $g(\vartheta_k;\vartheta_k) = f(\vartheta_k)$. The update rule for the surrogate function follows that $\vartheta_{k+1} = \arg\max_\vartheta g(\vartheta;\vartheta_k)$. And this results in $f(\vartheta_{k+1}) \geq g(\vartheta_{k+1};\vartheta_k) \geq f(\vartheta_k)$.

Recall the following function $\mathcal{L}(\vartheta;\vartheta_k)$ in the main paper.

$$\mathcal{L}(\vartheta;\vartheta_k) = \sum_{\tau\in\mathcal{T}} \mathbb{E}_{p(z|\mathcal{D}_\tau^T;\vartheta_k)} \left[\ln p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta) - \ln p(z|\mathcal{D}_\tau^T;\vartheta_k)\right] \tag{8.65}$$

The expectation operation in the k-th iteration step corresponds to $\texttt{E-step} : q_\phi(z|\mathcal{D}_\tau^C) = p_{\vartheta_k}(z|\mathcal{D}_\tau^C)$, while the maximization operation in the k-th iteration step updates the parameter as $\texttt{M-step} : \vartheta_{k+1} = \arg\max_\vartheta \mathcal{L}(\vartheta;\vartheta_k)$.

$$\underbrace{\ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_k)}_{\text{Model Evidence}} \underset{\texttt{E-step}}{=} \mathcal{L}(\vartheta_k;\vartheta_k) \underset{\texttt{M-step}}{\leq} \mathcal{L}(\vartheta_{k+1};\vartheta_k) \tag{8.66a}$$

$$\leq \mathcal{L}(\vartheta_{k+1};\vartheta_k) + D_{KL}[p(z|\mathcal{D}_\tau^T;\vartheta_k) \parallel p(z|\mathcal{D}_\tau^T;\vartheta_{k+1})] = \underbrace{\ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_{k+1})}_{\text{Model Evidence}} \quad \square \tag{8.66b}$$

Proof of Improvement Guarantee using Variational EM

As illustrated in Eq.s (8.66), the surrogate function $\mathcal{L}(\vartheta;\vartheta_k)$ is bounded by two log-likelihoods. Over the process of iterations, the log-likelihood is gradually increased to the final convergence.

$$\begin{aligned}\ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_1) \leq \mathcal{L}(\vartheta_2;\vartheta_1) \leq \ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_2) \leq \cdots \\ \leq \mathcal{L}(\vartheta_H;\vartheta_{H-1}) \leq \ln p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_H)\end{aligned} \tag{8.67}$$

This indicates that the finally updated parameters of the surrogate function are exactly optimal ones for the evidence. Based on these rules, directly optimizing the surrogate function step by step can guarantee the finding of optimal parameters in a theoretical sense.

Importance Sampling in a Variational EM Algorithm

Though the conditional marginal distribution $p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_k)$ is not analytical, the importance sampling trick with help of a proposal distribution $q_\eta(z|\mathcal{D}_\tau^T)$ can be used to estimate the result.

$$
\begin{aligned}
p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_k) &\approx \frac{1}{B}\sum_{b=1}^{B}\omega^{(b)}, \\
\text{with } z^{(b)} \sim q_\eta(z|\mathcal{D}_\tau^T) \text{ and } \omega^{(b)} &= \frac{p(\mathcal{D}_\tau^T, z^{(b)}|\mathcal{D}_\tau^C;\vartheta_k)}{q_\eta(z^{(b)}|\mathcal{D}_\tau^T)}
\end{aligned} \tag{8.68}
$$

Especially, the joint distribution is computed via the decomposition that $p(\mathcal{D}_\tau^T, z^{(b)}|\mathcal{D}_\tau^C;\vartheta_k) = p(z^{(b)}|\mathcal{D}_\tau^C;\vartheta_k)p(\mathcal{D}_\tau^T|z^{(b)};\vartheta_k)$ with $p(\mathcal{D}_\tau^T|z^{(b)};\vartheta_k) = \prod_{i=1}^{n+m} p(y_i|[x_i, z^{(b)}];\vartheta_k)$.

With the above equation, the intractable optimization objective is transformed into a feasible one.

$$
\begin{aligned}
\mathcal{L}_{\text{EM}}(\vartheta;\vartheta_k) &= \mathbb{E}_{p(z|\mathcal{D}_\tau^T;\vartheta_k)} \ln p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta) && \text{(8.69a)} \\
&= \int \frac{p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta_k)}{p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_k)} \ln p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta)dz && \text{(8.69b)} \\
&= \int q_\eta(z|\mathcal{D}_\tau^T)\frac{p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta_k)}{q_\eta(z|\mathcal{D}_\tau^T)p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_k)} \ln p(\mathcal{D}_\tau^T, z|\mathcal{D}_\tau^C;\vartheta)dz && \text{(8.69c)} \\
&\approx \frac{1}{B}\sum_{b=1}^{B}\frac{\omega^{(b)}}{p(\mathcal{D}_\tau^T|\mathcal{D}_\tau^C;\vartheta_k)} \ln p(\mathcal{D}_\tau^T, z^{(b)}|\mathcal{D}_\tau^C;\vartheta) && \text{(8.69d)} \\
&= \sum_{b=1}^{B}\frac{\omega^{(b)}}{\sum_{b'=1}^{B}\omega^{(b')}} \ln p(\mathcal{D}_\tau^T, z^{(b)}|\mathcal{D}_\tau^C;\vartheta) && \text{(8.69e)} \\
&= \sum_{b=1}^{B}\hat{\omega}^{(b)} \ln p(\mathcal{D}_\tau^T, z^{(b)}|\mathcal{D}_\tau^C;\vartheta) && \text{(8.69f)} \\
&= \mathcal{L}_{\text{SI-NP}}(\vartheta, \eta;\vartheta_k) && \text{(8.69g)}
\end{aligned}
$$

We can expand the term inside the expectation in Eq (8.69.a) as follows.

$$
p(\mathcal{D}_\tau^T, z^{(b)}|\mathcal{D}_\tau^C;\vartheta) = p(z^{(b)}|\mathcal{D}_\tau^C;\vartheta)p(\mathcal{D}_\tau^T|z^{(b)};\vartheta) \tag{8.70}
$$

As for the posterior $p(z|\mathcal{D}_\tau^T;\vartheta_k)$, we can get the following expansion.

$$
p(z|\mathcal{D}_\tau^T;\vartheta_k) = \frac{p(z, \mathcal{D}_\tau^T;\vartheta_k)}{\int p(z, \mathcal{D}_\tau^T;\vartheta_k)dz} = \frac{p(z|\mathcal{D}_\tau^C;\vartheta_k)p(\mathcal{D}_\tau^T|z;\vartheta_k)}{\int p(z|\mathcal{D}_\tau^C;\vartheta_k)p(\mathcal{D}_\tau^T|z;\vartheta_k)dz} \tag{8.71}
$$

The distribution is with the complicated denominator and this makes it infeasible to directly sample from the conditional distribution.

Optimization Objective with Proposal Distributions

It is trivial to see the following equation.

$$\begin{aligned} &\arg\min_{\eta} D_{KL}[p(z|\mathcal{D}_\tau^T;\vartheta_k) \parallel q_\eta(z|\mathcal{D}_\tau^T)] \\ \Leftrightarrow &\arg\min_{\eta} -\mathbb{E}_{p(z|\mathcal{D}_\tau^T;\vartheta_k)}[\ln q_\eta(z|\mathcal{D}_\tau^T)] \end{aligned} \tag{8.72}$$

Once again, we apply self-normalized importance sampling to the right side of Eq. (8.72). With the same set of sampled latent variables, the reweighted objective *w.r.t.* the proposal distribution can be derived.

$$\begin{aligned} &\arg\min_{\eta} -\mathbb{E}_{p(z|\mathcal{D}_\tau^T;\vartheta_k)}[\ln q_\eta(z|\mathcal{D}_\tau^T)] \\ \approx &-\sum_{b=1}^{B} \hat{\omega}^{(b)} \ln q_\eta(z^{(b)}|\mathcal{D}_\tau^T) = \mathcal{L}_{\mathrm{KL}}(\eta;\eta_{k-1},\vartheta_k) \end{aligned} \tag{8.73}$$

To include the inequality to show the bias in importance sampling and the decrease of the bias with more samples and morel updates.

Gradient Estimates in Variational EM

In the `E-step` #2, note that the model parameter is fixed as ϑ_k, we can estimate the gradient of η *w.r.t.* $\mathcal{L}_{\mathrm{KL}}(\eta;\eta_{k-1},\vartheta_k)$ in the following way.

$$\frac{\partial \mathcal{L}_{\mathrm{KL}}(\eta;\eta_{k-1},\vartheta_k)}{\partial \eta} = \sum_{b=1}^{B} \hat{\omega}^{(b)} \left(\frac{\partial \ln q_\eta(z^{(b)}|\mathcal{D}_\tau^T)}{\partial \eta} \right) \tag{8.74}$$

In the `M-step`, note that the normalized importance weights are constant, the proposal distribution is fixed, and the gradient *w.r.t.* $\mathcal{L}_{\text{SI-NP}}(\vartheta;\eta_k,\vartheta_k)$ can be estimated in a straightforward way as follows.

$$\frac{\partial \mathcal{L}_{\text{SI-NP}}(\vartheta;\eta_k,\vartheta_k)}{\partial \vartheta} = \sum_{b=1}^{B} \hat{\omega}^{(b)} \left(\frac{\partial \ln p(z^{(b)}|\mathcal{D}_\tau^C;\vartheta)}{\partial \vartheta} + \frac{\partial \ln p(\mathcal{D}_\tau^T|z^{(b)};\vartheta)}{\partial \vartheta} \right) \tag{8.75}$$

D.6 *Proof of SI-NPs Equivalence with CNPs*

Prior Collapse in SI-NPs with One Monte Carlo Sample

Theorem 6 (L'Hôpital's Rule (Hospital, 1696)) *Let $f(x)$ and $g(x)$ be two functions differentiable on an open interval I except possibly at a point c contained in I. If*

$\lim_{x\to c} f(x) = \lim_{x\to c} g(x) = \infty$, *and* $\left(\frac{1}{g(x)f(x)}\right)' \neq 0$ *with* $\forall x \in \mathcal{I}$ *and* $x \neq c$, *we can have the following limit equation.*

$$\lim_{x\to c} f(x) - g(x) = \lim_{x\to c} \frac{\frac{1}{g(x)} - \frac{1}{f(x)}}{\frac{1}{g(x)f(x)}} = \lim_{x\to c} \frac{\left(\frac{1}{g(x)} - \frac{1}{f(x)}\right)'}{\left(\frac{1}{g(x)f(x)}\right)'} \tag{8.76}$$

Proof 2 (Proposition 3) *Let* $z \in \mathbb{R}^d$ *be the latent variable for a diagonal Gaussian conditional prior* $p(z|\mathcal{D}_\tau^C;\vartheta) = \mathcal{N}(z;\mu_\vartheta(\mathcal{D}_\tau^C),\Sigma_\vartheta(\mathcal{D}_\tau^C))$. *Here the learned mean and the covariance matrix are simply denoted by* $\mu_\vartheta = [\mu_1,\ldots,\mu_d]^T \in \mathbb{R}^d$ *and* $\Sigma_\vartheta = diag\left[\sigma_1^2,\ldots,\sigma_d^2\right]$.

With one Monte Carlo sample $\hat{z} = [\hat{z}_1,\ldots,\hat{z}_d]^T$ *from the conditional prior, it can be written as* $\hat{z} = \mu_\vartheta + \hat{\epsilon}\Sigma_\vartheta^{\frac{1}{2}}, \hat{\epsilon} \sim \mathcal{N}(0,\mathcal{I}_d)$ *with help of a reparameterization trick (Kingma and Welling, 2013).*

$$\mathbb{E}_{p(z|\mathcal{D}_\tau^C;\vartheta_k)}\left[\ln p(z|\mathcal{D}_\tau^C;\vartheta)\right] \approx \ln p(z|\mathcal{D}_\tau^C;\vartheta) = -\frac{1}{2}\ln(2\pi) + \sum_{i=1}^{d}\left[-\ln\sigma_i - \frac{(\mu_i-\hat{z}_i)^2}{2\sigma_i^2}\right] \tag{8.77}$$

Eq. (8.77) is the result of one Monte Carlo estimate, termed as the collapse term in the main paper. Built up on these, we rewrite the SI-NP optimization objective to maximize as Eq. (8.78).

$$\begin{aligned}\mathcal{L}_{SI\text{-}NP} &= \mathbb{E}_{p(z|\mathcal{D}_\tau^C;\vartheta)}\left[\ln p(\mathcal{D}_\tau^T|z;\vartheta)\right] + \mathbb{E}_{p(z|\mathcal{D}_\tau^C;\vartheta_k)}\left[\ln p(z|\mathcal{D}_\tau^C;\vartheta)\right] \\ &\approx \sum_{i=1}^{n+m}\ln p(y_i|[x_i,\mu_\vartheta + \hat{\epsilon}\Sigma_\vartheta^{\frac{1}{2}};\vartheta]) - \left(\frac{1}{2}\ln(2\pi) + \sum_{i=1}^{d}\left[\ln\sigma_i + \frac{(\mu_i-\hat{z}_i)^2}{2\sigma_i^2}\right]\right)\end{aligned} \tag{8.78}$$

Now we prove that when the learned variance parameter $\{\sigma_i\}_{i=1}^d$ *collapse into the value zero, the optimization objective in Eq. (8.77) and Eq. (8.78) can be maximized. To simplify the notation, we put the mean variable* μ_ϑ *aside, focus more on the variance variable* Σ_ϑ *and let the value* $\frac{\mu_i-\hat{z}_i}{2}$ *denoted by* κ_i.

$$\begin{aligned}\lim_{\sigma_i\to 0} -\ln\sigma_i - \frac{\kappa_i}{\sigma_i^2} &= \lim_{\sigma_i\to 0}\frac{\frac{\sigma_i^2}{\kappa_i} + \frac{1}{\ln\sigma_i}}{-\frac{\sigma_i^2}{\kappa_i\ln\sigma_i}} = \lim_{\sigma_i\to 0}\frac{\sigma_i^2\ln\sigma_i + \kappa_i}{-\sigma_i^2} \\ = \lim_{\sigma_i\to 0}\frac{\left(\sigma_i^2\ln\sigma_i + \kappa_i\right)'}{\left(-\sigma_i^2\right)'} &= \lim_{\sigma_i\to 0}\frac{2\sigma_i\ln\sigma_i + \sigma_i}{-2\sigma_i} = \lim_{\sigma_i\to 0} -\ln\sigma_i - \frac{1}{2} = +\infty\end{aligned} \tag{8.79}$$

Putting them together, the Gaussian latent variable will finally collapse into a Dirac delta distribution and this demonstrates the equivalence between SI-NP with one Monte Carlo sample and CNP.

Remark 2 *Either increasing the number of Monte Carlo samples or lower the weight of the collapse term can effectively avoid the prior collapse.*

With increase of Monte Carlo samples, we can see the scale of the generative term outweights that of the collapse term, which indicates more weights are put in the gradient *w.r.t.* the variance parameters to maximize the generative log-likelihood. In this way, it can naturally avoid the prior collapse caused by the second term.

D.7 *Experimental Setup & Implementation Details*

In all experiments, we use Adam (Kingma and Welling, 2013) as the default optimizer for all experiments. Pytorch works as the toolkit to program and run experiments.

Meta Learning Datasets

Gaussian Process Simulator. The generator of Gaussian process dataset is the same with that in (Lee et al., 2020a). Three types of kernels are used to formulate diverse Gaussian processes.

- Matern $-\frac{5}{2}$ kernel:

$$k(x,x') = \left(1 + 4\sqrt{5}d + \frac{5}{3}d^2\right)\exp\left(-\sqrt{5}d\right)$$

  with $d = 4|x - x'|$;

- RBF kernel:

$$k(x,x') = s^2 \exp\left(-\frac{(x-x')^2}{2l^2}\right)$$

  with $s \sim U[0.1, 1.0]$ and $l \sim U[0.1, 0.6]$

- Periodic kernel:

$$k(x,x') = s^2 \exp\left(\frac{-2\sin^2(\frac{\pi\|x-x'\|^2}{p})}{l^2}\right)$$

  with $s \sim U[0.1, 1.0]$, $l \sim U[0.1, 0.6]$ and $p \sim U[0.1, 0.5]$

Image Datasets. Benchmark image datasets include MNIST (Bottou et al., 1994), FMNIST (Xiao et al., 2017), CIFAR10 (Krizhevsky et al., 2009) and SVHN (Sermanet et al., 2012). For pixel values, they are transformed to normalized Tensors via pytorch package.

Neural Architectures & Optimizations

Synthetic Regression. In terms of neural architectures, we use the same setup as that in (Gordon et al., 2019; Lee et al., 2020a) for all baselines. The dimension of latent

variables is 128. The `Encoder` is a two hidden layer neural network with 128 neuron units for each layer. The `Decoder` is a one hidden layer neural network with 128 neuron units. The optimizer's learning rate is $5e-4$. For ML-NPs and SI-NPs, we use 16 Monte Carlo samples for latent variables in meta training, which is the default setting in (Foong et al., 2020).

Image Completion. The setup is the same with that in (Garnelo et al., 2018a) and works for all NPs variants. As default, we set the dimension of latent variables z as 128 for all baselines. For all baselines, the `Encoder` is constituted with three hidden layers (128 neuron units each). The `Decoder` has five hidden layers (128 neuron units each) as well. The learning rate for the optimizer is $5e-4$. The training batch size for all images is 4 and we meta train the model until convergence (the maximum epoch number for MNIST/FMNIST is 100, and that for CIFAR10/SVHN is 200, and early stop is used when necessary). For ML-NPs, we use 16 Monte Carlo samples for latent variables in meta training, which is the default setting in (Foong et al., 2020). For SI-NPs, we find that 8 Monte Carlo samples are sufficient in implementations.

BIBLIOGRAPHY

Allen, K., Shelhamer, E., Shin, H., and Tenenbaum, J. (2019). Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, pages 232–241. PMLR.

Amos, B., Dinh, L., Cabi, S., Rothörl, T., Colmenarejo, S. G., Muldal, A., Erez, T., Tassa, Y., de Freitas, N., and Denil, M. (2018). Learning awareness models. *arXiv preprint arXiv:1804.06318*.

Asmuth, J., Li, L., Littman, M. L., Nouri, A., and Wingate, D. (2009). A bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 19–26.

Bhattacharya, R. N. and Waymire, E. C. (2009). *Stochastic processes with applications*. SIAM.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Bonilla, E. V., Chai, K. M., and Williams, C. (2008). Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, pages 153–160.

Bornschein, J. and Bengio, Y. (2014). Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., et al. (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, volume 2, pages 77–82. IEEE.

Burt, D., Rasmussen, C. E., and Van Der Wilk, M. (2019). Rates of convergence for sparse variational gaussian process regression. In *International Conference on Machine Learning*, pages 862–871.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765.

Clavera, I., Nagabandi, A., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. (2019). Learning to adapt: Meta-learning for model-based control. *arXiv preprint arXiv:1803.11347*, 3.

Cremer, C., Li, X., and Duvenaud, D. (2018). Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR.

Dai, Z., Damianou, A. C., González, J., and Lawrence, N. D. (2016a). Variational auto-encoded deep gaussian processes. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016*.

Dai, Z., Damianou, A. C., González, J., and Lawrence, N. D. (2016b). Variational auto-encoded deep gaussian processes. In *International Conference on Learning Representations*.

Daumé III, H. (2009). Bayesian multitask learning with latent hierarchies. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 135–142.

De Finetti, B. (1937). La prévision: ses lois logiques, ses sources subjectives. In *Annales de l'institut Henri Poincaré*, volume 7, pages 1–68.

Deisenroth, M. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 465–472.

Denton, E. and Fergus, R. (2018). Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1174–1183.

Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F., and Udluft, S. (2018). Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1184–1193.

Dezfouli, A. and Bonilla, E. V. (2015). Scalable inference for gaussian process models with black-box likelihoods. In *Advances in Neural Information Processing Systems*, pages 1414–1422.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., and Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016). Rl ̃2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

Džeroski, S., Demšar, D., and Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, 13(1):7–17.

Edwards, H. and Storkey, A. (2017). Towards a neural statistician. *International Conference on Learning Representations*.

Eslami, S., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. (2016). Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems*, 29.

Eslami, S. A., Rezende, D. J., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., et al. (2018). Neural scene representation and rendering. *Science*, 360(6394):1204–1210.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Finn, C., Xu, K., and Levine, S. (2018). Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*.

Finzi, M., Welling, M., and Wilson, A. G. (2021). A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *International Conference on Machine Learning*.

Flennerhag, S., Rusu, A. A., Pascanu, R., Yin, H., and Hadsell, R. (2019). Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*.

Foong, A. Y., Bruinsma, W., Gordon, J., Dubois, Y., Requeima, J., and Turner, R. E. (2020). Meta-learning stationary stochastic process prediction with convolutional neural processes. In *Advances in Neural Information Processing Systems*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Gal, Y., McAllister, R., and Rasmussen, C. E. (2016). Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, page 34.

Galashov, A., Schwarz, J., Kim, H., Garnelo, M., Saxton, D., Kohli, P., Eslami, S., and Teh, Y. W. (2019). Meta-learning surrogate models for sequential decision making. *arXiv preprint arXiv:1903.11907*.

Gao, A., Castellanos, J., Yue, Y., Ross, Z., and Bouman, K. (2021). Deepgem: Generalized expectation-maximization for blind inversion. *Advances in Neural Information Processing Systems*, 34:11592–11603.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. (2018a). Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. (2018b). Neural processes. *arXiv preprint arXiv:1807.01622*.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.

Ghojogh, B., Ghodsi, A., Karray, F., and Crowley, M. (2021). Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey. *arXiv preprint arXiv:2101.00734*.

Gondal, M. W., Joshi, S., Rahaman, N., Bauer, S., Wuthrich, M., and Schölkopf, B. (2021). Function contrastive learning of transferable meta-representations. In *International Conference on Machine Learning*, pages 3755–3765. PMLR.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. (2018). Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*.

Gordon, J., Bruinsma, W. P., Foong, A. Y., Requeima, J., Dubois, Y., and Turner, R. E. (2019). Convolutional conditional neural processes. In *International Conference on Learning Representations*.

Gordon, J., Bruinsma, W. P., Foong, A. Y., Requeima, J., Dubois, Y., and Turner, R. E. (2020a). Convolutional conditional neural processes. In *International Conference on Learning Representations*.

Gordon, J., Bruinsma, W. P., Foong, A. Y. K., Requeima, J., Dubois, Y., and Turner, R. E. (2020b). Convolutional conditional neural processes. In *8th International Conference on Learning Representations, ICLR 2020*.

Gumbel, E. J. (1954). *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. (2018). Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2018). Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. (2018). Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015). Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2(5):6.

Hiraoka, T., Imagawa, T., Tangkaratt, V., Osa, T., Onishi, T., and Tsuruoka, Y. (2020). Meta-model-based meta-policy optimization. *arXiv preprint arXiv:2006.02608*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hofer, E., Kloos, M., Krzykacz-Hausmann, B., Peschke, J., and Woltereck, M. (2002). An approximate epistemic uncertainty analysis approach in the presence of epistemic and aleatory uncertainties. *Reliability Engineering & System Safety*, 77(3):229–238.

Holderrieth, P., Hutchinson, M. J., and Teh, Y. W. (2021). Equivariant learning of stochastic fields: Gaussian processes and steerable conditional neural processes. In *International Conference on Machine Learning*, pages 4297–4307. PMLR.

Hospital, L. (1696). *Analyse des infiniment petits*.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Teh, Y. W., and Heess, N. (2019). Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*.

Hunter, D. R. and Lange, K. (2004). A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37.

Iakovleva, E., Verbeek, J., and Alahari, K. (2020). Meta-learning with shared amortized variational inference. In *International Conference on Machine Learning*, pages 4572–4582. PMLR.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*.

Jing, M., Ma, X., Sun, F., and Liu, H. (2018). Learning and inferring movement with deep generative model. *arXiv preprint arXiv:1805.07252*.

Kamthe, S. and Deisenroth, M. P. (2017). Data-efficient reinforcement learning with probabilistic model predictive control. *arXiv preprint arXiv:1706.06491*.

Kawano, M., Kumagai, W., Sannai, A., Iwasawa, Y., and Matsuo, Y. (2020). Group equivariant conditional neural processes. In *International Conference on Learning Representations*.

Kégl, B., Hurtado, G., and Thomas, A. (2021). Model-based micro-data reinforcement learning: what are the crucial model properties and which model to choose? *arXiv preprint arXiv:2107.11587*.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. (2020). Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707.

Killian, T. W., Daulton, S., Konidaris, G., and Doshi-Velez, F. (2017). Robust and efficient transfer learning with hidden parameter markov decision processes. *Advances in Neural Information Processing Systems*, 30.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2018). Attentive neural processes. In *International Conference on Learning Representations*.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2019a). Attentive neural processes. In *International Conference on Learning Representations*.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, S. M. A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2019b). Attentive neural processes. In *7th International Conference on Learning Representations, ICLR 2019*.

Kim, M., Go, K. R., and Yun, S.-Y. (2021). Neural processes with stochastic attention: Paying more attention to the context dataset. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2014). Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*.

Kohl, S., Romera-Paredes, B., Meyer, C., De Fauw, J., Ledsam, J. R., Maier-Hein, K., Eslami, S. A., Rezende, D. J., and Ronneberger, O. (2018). A probabilistic u-net for segmentation of ambiguous images. In *Advances in Neural Information Processing Systems*, pages 6965–6975.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

Lawrence, N. D. and Platt, J. C. (2004). Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on Machine learning*, page 65.

Le, T. A., Kim, H., Garnelo, M., Rosenbaum, D., Schwarz, J., and Teh, Y. W. (2018). Empirical evaluation of neural process objectives. In *NeurIPS workshop on Bayesian Deep Learning*, page 71.

Le, T. A., Kosiorek, A. R., Siddharth, N., Teh, Y. W., and Wood, F. (2020). Revisiting reweighted wake-sleep for models with stochastic control flow. In *Uncertainty in Artificial Intelligence*, pages 1039–1049. PMLR.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lee, J., Lee, Y., Kim, J., Yang, E., Hwang, S. J., and Teh, Y. W. (2020a). Bootstrapping neural processes. *Advances in Neural Information Processing Systems*, 33:6606–6615.

Lee, K., Seo, Y., Lee, S., Lee, H., and Shin, J. (2020b). Context-aware dynamics model for generalization in model-based reinforcement learning. *International conference on machine learning*.

Lee, Y. and Choi, S. (2018). Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2927–2936.

Leibfried, F., Kushman, N., and Hofmann, K. (2016). A deep learning approach for joint video frame and reward prediction in atari games. *arXiv preprint arXiv:1611.07078*.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. (2020). Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Li, J., Vuong, Q., Liu, S., Liu, M., Ciosek, K., Ross, K., Christensen, H. I., and Su, H. (2019). Multi-task batch reinforcement learning with metric learning. *arXiv preprint arXiv:1909.11373*.

Li, L., Yang, R., and Luo, D. (2020). Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. *International Conference on Learning Representations*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

Lin, Z., Thomas, G., Yang, G., and Ma, T. (2020). Model-based adversarial meta-reinforcement learning. *Advances in Neural Information Processing Systems*, 33.

Louizos, C., Shi, X., Schutte, K., and Welling, M. (2019). The functional neural process. In *Advances in Neural Information Processing Systems*, pages 8743–8754.

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

Ma, C., Li, Y., and Hernandez-Lobato, J. M. (2019). Variational implicit processes. In *International Conference on Machine Learning*, pages 4222–4233.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. (2016). Auxiliary deep generative models. In *International Conference on Machine Learning*, pages 1445–1453.

Mathieu, E., Foster, A., and Teh, Y. W. (2021). On contrastive representations of stochastic processes. *arXiv preprint arXiv:2106.10052*.

Mendonca, R., Geng, X., Finn, C., and Levine, S. (2020). Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling. *arXiv preprint arXiv:2006.07178*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60.

Moreno-Muñoz, P., Artés, A., and Álvarez, M. (2018). Heterogeneous multi-output gaussian process prediction. In *Advances in Neural Information Processing Systems*, pages 6711–6720.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. (2019). Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*.

Nagabandi, A., Yang, G., Asmar, T., Kahn, G., Levine, S., and Fearing, R. S. (2017). Neural network dynamics models for control of under-actuated legged millirobots. *arXiv preprint arXiv:1711.05253*.

Nalisnick, E. T., Matsukawa, A., Teh, Y. W., Görür, D., and Lakshminarayanan, B. (2019). Do deep generative models know what they don't know? In *7th International Conference on Learning Representations, ICLR 2019*.

Oksendal, B. (2013). *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016a). Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, pages 4026–4034.

Osband, I., Russo, D., and Van Roy, B. (2013). (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*.

Osband, I., Van Roy, B., and Wen, Z. (2016b). Generalization and exploration via randomized value functions. In *International Conference on Machine Learning*, pages 2377–2386. PMLR.

Parmas, P., Rasmussen, C. E., Peters, J., and Doya, K. (2018). Pipps: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, pages 4065–4074. PMLR.

Pei, J., Ren, P., Monz, C., and de Rijke, M. (2020). Retrospective and prospective mixture-of-generators for task-oriented dialogue response generation. In *ECAI 2020*, pages 2148–2155. IOS Press.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pertsch, K., Lee, Y., and Lim, J. J. (2020). Accelerating reinforcement learning with learned skill priors. *Conference on Robot Learning*.

Rajbhandari, S., Li, C., Yao, Z., Zhang, M., Aminabadi, R. Y., Awan, A. A., Rasley, J., and He, Y. (2022). Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. *arXiv preprint arXiv:2201.05596*.

Rajeswaran, A., Finn, C., Kakade, S., and Levine, S. (2019). Meta-learning with implicit gradients. *Advances in Neural Information Processing Systems*.

Rajeswaran, A., Mordatch, I., and Kumar, V. (2020). A game theoretic framework for model based reinforcement learning. *International conference on machine learning*.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. (2019). Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer.

Rasmussen, C. E. and Ghahramani, Z. (2002). Infinite mixtures of gaussian process experts. *Advances in Neural Information Processing Systems*, 2:881–888.

Ren, H., Garg, A., and Anandkumar, A. (2019). Contextbased meta-reinforcement learning with structured latent space. In *Skills Workshop NeurIPS*.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. (2019). Fast and flexible multi-task classification using conditional neural adaptive processes. *Advances in Neural Information Processing Systems*, 32.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR.

Ross, S. M., Kelly, J. J., Sullivan, R. J., Perry, W. J., Mercer, D., Davis, R. M., Washburn, T. D., Sager, E. V., Boyce, J. B., and Bristow, V. L. (1996). *Stochastic processes*, volume 2. Wiley New York.

Rudner, T. G., Fortuin, V., Teh, Y. W., and Gal, Y. (2018). On the connection between neural processes and gaussian processes with deep kernels. In *Workshop on Bayesian Deep Learning, NeurIPS*.

Russo, D. and Van Roy, B. (2014). Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243.

Sæmundsson, S., Hofmann, K., and Deisenroth, M. P. (2018). Meta reinforcement learning with latent variable gaussian processes. *arXiv preprint arXiv:1803.07551*.

Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4588–4599.

Salimbeni, H., Dutordoir, V., Hensman, J., and Deisenroth, M. (2019). Deep gaussian processes with importance-weighted variational inference. In *International Conference on Machine Learning*, pages 5589–5598.

Satorras, V. G. and Estrach, J. B. (2018). Few-shot learning with graph neural networks. In *International Conference on Learning Representations*.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR.

Schirmer, M., Eltayeb, M., Lessmann, S., and Rudolph, M. (2022). Modeling irregular time series with continuous recurrent units. In *International Conference on Machine Learning*, pages 19388–19405. PMLR.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sermanet, P., Chintala, S., and LeCun, Y. (2012). Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 3288–3291. IEEE.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2016). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

Shi, Y., Paige, B., Torr, P., et al. (2019). Variational mixture-of-experts autoencoders for multi-modal deep generative models. *Advances in Neural Information Processing Systems*, 32.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Singh, G., Yoon, J., Son, Y., and Ahn, S. (2019). Sequential neural processes. In *Advances in Neural Information Processing Systems*, pages 10254–10264.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090.

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264.

Snelson, E., Ghahramani, Z., and Rasmussen, C. E. (2004). Warped gaussian processes. In *Advances in Neural Information Processing Systems*, pages 337–344.

Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). How to train deep variational autoencoders and probabilistic ladder networks. In *33rd International Conference on Machine Learning (ICML 2016)*.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., and Vlahavas, I. (2016). Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, 104(1):55–98.

Sun, S., Zhang, G., Shi, J., and Grosse, R. B. (2019). Functional variational bayesian neural networks. In *7th International Conference on Learning Representations, ICLR 2019*.

Sun, Z., Wu, J., Li, X., Yang, W., and Xue, J.-H. (2021). Amortized bayesian prototype meta-learning: A new probabilistic meta-learning approach to few-shot image classification. In *International Conference on Artificial Intelligence and Statistics*, pages 1414–1422. PMLR.

Sutton, R. (2019). The bitter lesson. *Incomplete Ideas (blog)*, 13:12.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476):1566–1581.

Thakur, S., van Hoof, H., Higuera, J. C. G., Precup, D., and Meger, D. (2019). Uncertainty aware learning from demonstrations in multiple contexts using bayesian neural networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 768–774. IEEE.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574.

Titsias, M. and Lawrence, N. D. (2010). Bayesian gaussian process latent variable model. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 844–851. JMLR Workshop and Conference Proceedings.

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational bayes for non-conjugate inference. In *International Conference on Machine Learning*, pages 1971–1979.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Tokdar, S. T. and Kass, R. E. (2010). Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60.

Tomczak, J. and Welling, M. (2018). Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR.

Tran, D., Ranganath, R., and Blei, D. (2017). Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5523–5533.

Tran, D., Ranganath, R., and Blei, D. M. (2016). Variational gaussian process. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016*.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

van der Pol, E., Kipf, T., Oliehoek, F. A., and Welling, M. (2020). Plannable approximations to mdp homomorphisms: Equivariance under actions. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1431–1439.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., et al. (2019). Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind Blog*.

Volpp, M., Flürenbrock, F., Grossberger, L., Daniel, C., and Neumann, G. (2020). Bayesian context aggregation for neural processes. In *International Conference on Learning Representations*.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

Wang, Q., Federici, M., and Van Hoof, H. (2022). Bridge the inference gaps of neural processes via expectation maximization. *In Preparation for International Conference on Learning Representations*.

Wang, Q. and Van Hoof, H. (2020). Doubly stochastic variational inference for neural processes with hierarchical latent variables. In *International Conference on Machine Learning*, pages 10018–10028. PMLR.

Wang, Q. and Van Hoof, H. (2022a). Learning expressive meta-representations with mixture of expert neural processes. *Advances in Neural Information Processing Systems*.

Wang, Q. and Van Hoof, H. (2022b). Model-based meta reinforcement learning using graph structured surrogate models and amortized policy search. In *International Conference on Machine Learning*, pages 23055–23077. PMLR.

Wang, T. and Ba, J. (2019). Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*.

Wang, T., Liao, R., Ba, J., and Fidler, S. (2018). Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*.

Waterhouse, S., MacKay, D., Robinson, T., et al. (1996). Bayesian methods for mixtures of experts. *Advances in Neural Information Processing Systems*, pages 351–357.

Welling, M. (2019). Do we still need models or just more data and compute?

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016a). Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. (2016b). Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, L., Jordan, M. I., and Hinton, G. E. (1995). An alternative model for mixtures of experts. *Advances in Neural Information Processing Systems*, pages 633–640.

Yaqoob, I., Khan, L. U., Kazmi, S. A., Imran, M., Guizani, N., and Hong, C. S. (2019). Autonomous driving cars in smart cities: Recent advances, requirements, and challenges. *IEEE Network*, 34(1):174–181.

Yoon, J., Singh, G., and Ahn, S. (2020). Robustifying sequential neural processes. In *International Conference on Machine Learning*, pages 10861–10870. PMLR.

Yuksel, S. E., Wilson, J. N., and Gader, P. D. (2012). Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. (2018). Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026.

Zhao, J. and Sun, S. (2016). Variational dependent multi-output gaussian process dynamical systems. *The Journal of Machine Learning Research*, 17(1):4134–4169.

Zimmermann, H., Wu, H., Esmaeili, B., and van de Meent, J.-W. (2021). Nested variational inference. *Advances in Neural Information Processing Systems*, 34.

Zintgraf, L., Shiarli, K., Kurin, V., Hofmann, K., and Whiteson, S. (2019). Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702. PMLR.

SAMENVATTING

Dit proefschrift richt zich op het leren van functie representaties voor het kwantificeren van onzekerheid & de snelle overdracht van vaardigheden in diepe neurale netwerken (DNNs). Om DNNs te gebruiken in realistische, praktische toepassingen, zijn vooralsnog complexe eigenschappen nodig. Dit project richt zich op twee van deze eigenschappen, namelijk het kwantificeren van onzekerheid en snelle overdracht van vaardigheden in DNNs. De eerste eigenschap maakt het mogelijk om risico-gevoelige beslissingen te kunnen nemen met behulp van DNNs, en kan mogelijk zorgen dat er efficiënter geleerd kan worden met minder datapunten. De tweede eigenschap voorkomt dat diepe netwerken zonder enige voorkennis getraind moeten worden, en kan er daarnaast voor zorgen dat DNNs zich makkelijker kunnen aanpassen aan nieuwe taken.

Mijn stelling is dat het gebruik van functie representaties een tastbare oplossing kan bieden voor het implementeren van de bovengenoemde eigenschappen. In plaats van elk datapunt te representeren met variational autoencoders (Kingma and Welling, 2013), kan een gehele verzameling datapunten gerepresenteerd worden door middel van een functie. In deze thesis wordt een functie representatie gedefinieerd als een samenvatting van context datapunten $\mathcal{D}^C = \{(x_i, y_i)\}_{i=1}^n$ voor het representeren van hun onderliggende functie $y = f(x)$.

Een typisch voorbeeld is een standaard neuraal proces (NP) (Garnelo et al., 2018b), waarin door middel van een benaderende a-priori-functie $q_\phi(z|\mathcal{D}^C)$ de voorspellende kansverdeling $\mathbb{E}_{q_\phi(z|\mathcal{D}^C)}\left[p(y|z,x)\right]$ wordt geïnduceerd. Op vergelijkbare wijze zal in dit proefschrift de standaard NP worden gebruikt als functie representatie model, waarop de algoritmen in dit proefschrift zullen voortbouwen. Een groot deel van dit proefschrift beschrijft hoe structurele inductieve biases kunnen worden meegenomen in functie representaties. Daarnaast zal een analyse worden gegeven van huidige methoden voor het optimaliseren van standaard NPs, en nieuwe manieren worden voorgesteld die het probleem van de inference gap hierin aanpakken.

De wetenschappelijke bijdragen van dit proefschrift zijn de volgende:

- In hoofdstuk (3) wordt een Doubly Stochastic Variational Neural Process (DSVNP) (Wang and Van Hoof, 2020) beschreven voor multi-output regressie en onzekerheidskwantificatie in beeldclassificatie. DSNVP is een hiërarchisch Bayesiaans model dat zowel functies representeerd als correlaties tussen invoer- en uitvoerdata meet. Experimentele resultaten laten zien dat DSVNP competitief presteert tegenover huidige SOTA modellen als het gaat om het modelleren van hoog-dimensionale realistische datasets en het kwantificeren van onzekerheid in de beeldclassificatie voorspellingen.

- In hoofdstuk (4) wordt een Mixture-of-Expert Neural Process (MoE-NP) (Wang and Van Hoof, 2022a) voor few-shot regressie en meta reinforcement learning problemen beschreven. MoE-NP is een Bayesiaans mixture model dat gebruik maakt van zowel continue latente variabelen als categorische latente variabelen, die het samen mogelijk maken om te werken met mixes van diverse componenten en taken in multi-modale kansverdelingen. Experimentele resultaten laten zien dat MoE-NP onderscheid kan maken tussen verschillende van deze functie componenten door middel van latente variabelen, en behaald daarnaast SOTA prestaties op CIFAR-10 beeldvoltooiing en continue sturingstaken.

- Hoofdstuk (5) stelt het Graph Structured Surrogate Model (GSSM) in combinatie met Amortized Policy Search (APS) (Wang and Van Hoof, 2022b) voor, welke van toepassing is voor Model-based meta reinforcement learning. Deze methode leert tegelijkertijd de dynamiekmodellen als optimale policies in model-based meta reinforcement learning taken. Zelfs wanneer het aantal training episodes gelimiteerd is, kunnen de geleerde dynamiekmodellen, welke gebruik maken van latente variabelen met graafstrukturen, goed generaliseren naar nieuwe MDPs, terwijl de geleerde globale latente variabele taak-specifieke value functions en policies induceert. Deze methoden leveren hoge prestaties op de Cartpole/Acrobot en Mujoco environments.
- Hoofdstuk (6) geeft een analyse van de huidige optimalisatiedoelen van NPs, analyseert hun suboptimale inferentie, en stelt hiervoor oplossingen voor die gebruik maken van variational expectation maximization (Wang et al., 2022). Het voorgestelde model, dat we het Self-normalized Importance weighted Neural Process (SI-NP) noemen, kan direct de effectieve a-priori functie aanleren zonder dat daarbij variational inference aan te pas hoeft te komen, en kan tevens garanties bieden dat de geleerde log-likelihood beter wordt over de functie dataset. Tegelijkertijd wordt gedemonstreerd dat het conditional neural process (CNP) (Garnelo et al., 2018a) equivalent is aan SI-NPs waarbij Monte-Carlo op basis van een enkele trekking wordt gebruikt. Deze claims worden geverifieerd op Gaussian process en beeldvoltooiing datasets.